



**EZ-PD™ CCGx Firmware Stack
API Reference Guide
Version 3.4**

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709
Phone (USA): 800.858.1810
Phone (Intl): 408.943.2600
<http://www.cypress.com>

Copyright © 2015-2020 Cypress Semiconductor Corporation. All rights reserved.

EZ-PD™ is a trademark of Cypress Semiconductor. All other trademarks or registered trademarks referenced herein are the property of their respective owners.

The information in this document is subject to change without notice and should not be construed as a commitment by Cypress. While reasonable precautions have been taken, Cypress assumes no responsibility for any errors that may appear in this document. No part of this document may be copied or reproduced in any form or by any means without the prior written consent of Cypress. Made in the U.S.A.

Disclaimer

CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

License Agreement

Please read the license agreement during SDK installation.

Contents

- 1 CCGx Firmware Stack: API Reference Guide 1**
 - 1.1 Introduction 1
 - 1.1.1 USB Type-C Highlights 1
 - 1.2 Cypress EZ-PD™ Type-C Controllers 1
 - 1.2.1 CCGx Product Families 1
 - 1.3 CCGx Firmware Stack 2
 - 1.3.1 Firmware Stack Organization 2

- 2 CCGx Firmware Architecture 5**
 - 2.1 CCGx Firmware Solution Structure 6
 - 2.2 Public API Summary 7
 - 2.2.1 Device Policy Manager (DPM) API 7
 - 2.2.2 Host Processor Interface (HPI) API 8
 - 2.2.3 Application Layer API 9
 - 2.2.3.1 Solution Layer Functions 10
 - 2.2.4 Alternate Mode Manager API 10
 - 2.2.5 Hardware Abstraction Layer (HAL) API 10
 - 2.2.5.1 GPIO API 10
 - 2.2.5.2 I2C Driver API 10
 - 2.2.5.3 Flash Module API 11
 - 2.2.5.4 Soft Timer API 11
 - 2.2.6 Firmware Update API 11

- 3 Data Structure Index 13**
 - 3.1 Data Structures 13

- 4 File Index 19**
 - 4.1 File List 19

- 5 Data Structure Documentation 21**
 - 5.1 pd_do_t::ACT_CBL_VDO Struct Reference 21
 - 5.1.1 Detailed Description 21
 - 5.1.2 Field Documentation 21

5.1.2.1	cbl_fw_ver	21
5.1.2.2	cbl_hw_ver	22
5.1.2.3	cbl_latency	22
5.1.2.4	cbl_term	22
5.1.2.5	max_vbus_volt	22
5.1.2.6	rsvd1	22
5.1.2.7	rsvd2	22
5.1.2.8	sop_dp	22
5.1.2.9	typec_abc	22
5.1.2.10	typec_plug	22
5.1.2.11	usb_ss_sup	23
5.1.2.12	vbus_cur	23
5.1.2.13	vbus_thru_cbl	23
5.1.2.14	vdo_version	23
5.2	pd_do_t::ACT_CBL_VDO_1 Struct Reference	23
5.2.1	Detailed Description	23
5.2.2	Field Documentation	24
5.2.2.1	cbl_fw_ver	24
5.2.2.2	cbl_hw_ver	24
5.2.2.3	cbl_latency	24
5.2.2.4	cbl_term	24
5.2.2.5	max_vbus_volt	24
5.2.2.6	rsvd1	24
5.2.2.7	rsvd2	24
5.2.2.8	sbu_supp	24
5.2.2.9	sbu_type	25
5.2.2.10	sop_dp	25
5.2.2.11	typec_abc	25
5.2.2.12	usb_ss_sup	25
5.2.2.13	vbus_cur	25
5.2.2.14	vbus_thru_cbl	25
5.2.2.15	vdo_version	25
5.3	pd_do_t::ACT_CBL_VDO_2 Struct Reference	25
5.3.1	Detailed Description	26
5.3.2	Field Documentation	26
5.3.2.1	active_el	26
5.3.2.2	max_op_temp	26
5.3.2.3	opt_isolated	26
5.3.2.4	phy_conn	26
5.3.2.5	rsvd0	26

5.3.2.6	rsvd1	27
5.3.2.7	shutdown_temp	27
5.3.2.8	ss_lanes	27
5.3.2.9	ss_supp	27
5.3.2.10	u3_power	27
5.3.2.11	u3_u0_trans	27
5.3.2.12	usb2_hub_hops	27
5.3.2.13	usb2_supp	27
5.3.2.14	usb4_supp	27
5.3.2.15	usb_gen	28
5.4	pd_do_t::ADO_ALERT Struct Reference	28
5.4.1	Detailed Description	28
5.4.2	Field Documentation	28
5.4.2.1	bat_status_change	28
5.4.2.2	fixed_bats	28
5.4.2.3	hot_swap_bats	28
5.4.2.4	ocp	29
5.4.2.5	op_cond_change	29
5.4.2.6	otp	29
5.4.2.7	ovp	29
5.4.2.8	rsvd1	29
5.4.2.9	rsvd2	29
5.4.2.10	src_input_change	29
5.5	alt_cfg_settings_t Struct Reference	29
5.5.1	Detailed Description	30
5.5.2	Field Documentation	30
5.5.2.1	dfp_mask	30
5.5.2.2	table_len	30
5.5.2.3	ufp_mask	30
5.6	alt_mode_evt_t::ALT_MODE_EVT Struct Reference	30
5.6.1	Detailed Description	30
5.6.2	Field Documentation	30
5.6.2.1	alt_mode	31
5.6.2.2	alt_mode_evt	31
5.6.2.3	data_role	31
5.6.2.4	svid	31
5.7	alt_mode_evt_t::ALT_MODE_EVT_DATA Struct Reference	31
5.7.1	Detailed Description	31
5.7.2	Field Documentation	31
5.7.2.1	evt_data	31

5.7.2.2	evt_type	32
5.8	alt_mode_evt_t Union Reference	32
5.8.1	Detailed Description	32
5.8.2	Field Documentation	32
5.8.2.1	alt_mode_event	32
5.8.2.2	alt_mode_event_data	32
5.8.2.3	val	32
5.9	alt_mode_hw_evt_t::ALT_MODE_HW_EVT Struct Reference	33
5.9.1	Detailed Description	33
5.9.2	Field Documentation	33
5.9.2.1	data_role	33
5.9.2.2	evt_data	33
5.9.2.3	hw_type	33
5.10	alt_mode_hw_evt_t Union Reference	33
5.10.1	Detailed Description	34
5.10.2	Field Documentation	34
5.10.2.1	hw_evt	34
5.10.2.2	val	34
5.11	alt_mode_info_t Struct Reference	34
5.11.1	Detailed Description	34
5.11.2	Field Documentation	35
5.11.2.1	alt_mode_id	35
5.11.2.2	app_evt_data	35
5.11.2.3	app_evt_needed	35
5.11.2.4	cbk	35
5.11.2.5	cbl_obj_pos	35
5.11.2.6	custom_att_obj_pos	35
5.11.2.7	eval_app_cmd	35
5.11.2.8	is_active	35
5.11.2.9	mode_state	36
5.11.2.10	obj_pos	36
5.11.2.11	set_mux_isolate	36
5.11.2.12	sop_state	36
5.11.2.13	uvdm_supp	36
5.11.2.14	vdm_header	36
5.11.2.15	vdo	36
5.11.2.16	vdo_max_numb	36
5.11.2.17	vdo_numb	36
5.12	alt_mode_reg_info_t Struct Reference	37
5.12.1	Detailed Description	37

5.12.2	Field Documentation	37
5.12.2.1	alt_mode_id	37
5.12.2.2	app_evt	37
5.12.2.3	atch_tgt_info	37
5.12.2.4	atch_type	37
5.12.2.5	cbl_sop_flag	37
5.12.2.6	data_role	38
5.12.2.7	svid_emca_vdo	38
5.12.2.8	svid_vdo	38
5.13	app_cbk_t Struct Reference	38
5.13.1	Detailed Description	39
5.13.2	Field Documentation	39
5.13.2.1	app_event_handler	39
5.13.2.2	eval_dr_swap	39
5.13.2.3	eval_enter_usb	39
5.13.2.4	eval_fr_swap	39
5.13.2.5	eval_pr_swap	39
5.13.2.6	eval_rdo	39
5.13.2.7	eval_src_cap	39
5.13.2.8	eval_vconn_swap	40
5.13.2.9	eval_vdm	40
5.13.2.10	psnk_disable	40
5.13.2.11	psnk_enable	40
5.13.2.12	psnk_set_current	40
5.13.2.13	psnk_set_voltage	40
5.13.2.14	psrc_disable	40
5.13.2.15	psrc_enable	40
5.13.2.16	psrc_get_voltage	40
5.13.2.17	psrc_set_current	41
5.13.2.18	psrc_set_voltage	41
5.13.2.19	vbus_discharge_off	41
5.13.2.20	vbus_discharge_on	41
5.13.2.21	vbus_get_value	41
5.13.2.22	vbus_is_present	41
5.13.2.23	vconn_disable	41
5.13.2.24	vconn_enable	41
5.13.2.25	vconn_is_present	41
5.14	app_resp_t Struct Reference	42
5.14.1	Detailed Description	42
5.14.2	Field Documentation	42

5.14.2.1	req_status	42
5.14.2.2	resp_do	42
5.15	app_status_t Struct Reference	42
5.15.1	Detailed Description	43
5.15.2	Field Documentation	43
5.15.2.1	actv_swap_count	43
5.15.2.2	actv_swap_delay	44
5.15.2.3	actv_swap_type	44
5.15.2.4	alt_mode_entered	44
5.15.2.5	alt_mode_trig_mask	44
5.15.2.6	app_pending_swaps	44
5.15.2.7	app_resp	44
5.15.2.8	bc_12_src_disabled	44
5.15.2.9	cable_retimer_supp	44
5.15.2.10	cbl_disc_id_finished	44
5.15.2.11	cbl_rst_done	45
5.15.2.12	cur_fb_enabled	45
5.15.2.13	custom_hpi_svid	45
5.15.2.14	debug_acc_attached	45
5.15.2.15	dfp_alt_mode_mask	45
5.15.2.16	disc_cbl_pending	45
5.15.2.17	fault_status	45
5.15.2.18	is_hot_shutdown	45
5.15.2.19	is_mux_busy	45
5.15.2.20	is_vbus_on	46
5.15.2.21	is_vconn_on	46
5.15.2.22	is_vdm_pending	46
5.15.2.23	keep_vconn_src	46
5.15.2.24	ld_sw_ctrl	46
5.15.2.25	mux_poll_cbk	46
5.15.2.26	psnk_cur	46
5.15.2.27	psnk_volt	46
5.15.2.28	psrc_rising	46
5.15.2.29	psrc_volt	47
5.15.2.30	psrc_volt_old	47
5.15.2.31	pwr_ready_cbk	47
5.15.2.32	retimer_dis_req	47
5.15.2.33	skip_mux_config	47
5.15.2.34	snk_dis_cbk	47
5.15.2.35	trig_cbl_rst	47

5.15.2.36	turn_off_temp_limit	47
5.15.2.37	turn_on_temp_limit	47
5.15.2.38	ufp_alt_mode_mask	48
5.15.2.39	usb2_supp	48
5.15.2.40	usb3_supp	48
5.15.2.41	usb4_active	48
5.15.2.42	usb4_data_rst_cnt	48
5.15.2.43	vdm_prcs_failed	48
5.15.2.44	vdm_resp	48
5.15.2.45	vdm_resp_cbk	48
5.15.2.46	vdm_retry_pending	48
5.15.2.47	vdm_task_en	49
5.15.2.48	vdm_version	49
5.16	atch_tgt_info_t Struct Reference	49
5.16.1	Detailed Description	49
5.16.2	Field Documentation	49
5.16.2.1	ama_vdo	49
5.16.2.2	cbl_svid	49
5.16.2.3	cbl_vdo	49
5.16.2.4	tgt_id_header	50
5.16.2.5	tgt_svid	50
5.17	auto_cfg_settings_t Struct Reference	50
5.17.1	Detailed Description	50
5.17.2	Field Documentation	50
5.17.2.1	policy_mgr_en	50
5.17.2.2	port_pwr	50
5.17.2.3	pps_en	51
5.17.2.4	reserved_0	51
5.17.2.5	sensor_data	51
5.17.2.6	sys_pwr	51
5.17.2.7	table_len	51
5.17.2.8	unconstrained_pwr_en	51
5.17.2.9	vin_oc1	51
5.17.2.10	vin_oc2	51
5.17.2.11	vin_oc3	51
5.17.2.12	vin_throttling_ctrl	52
5.18	bat_chg_params_t Struct Reference	52
5.18.1	Detailed Description	52
5.18.2	Field Documentation	52
5.18.2.1	reserved_0	52

5.18.2.2	reserved_1	52
5.18.2.3	table_len	52
5.18.2.4	vbatt_cutoff_volt	53
5.18.2.5	vbatt_dischg_en_volt	53
5.18.2.6	vbatt_max_cur	53
5.18.2.7	vbatt_max_volt	53
5.19	pd_do_t::BAT_SNK Struct Reference	53
5.19.1	Detailed Description	53
5.19.2	Field Documentation	53
5.19.2.1	max_voltage	53
5.19.2.2	min_voltage	54
5.19.2.3	op_power	54
5.19.2.4	supply_type	54
5.20	pd_do_t::BAT_SRC Struct Reference	54
5.20.1	Detailed Description	54
5.20.2	Field Documentation	54
5.20.2.1	max_power	54
5.20.2.2	max_voltage	54
5.20.2.3	min_voltage	55
5.20.2.4	supply_type	55
5.21	bb_settings_t Struct Reference	55
5.21.1	Detailed Description	55
5.21.2	Field Documentation	55
5.21.2.1	bb_always_on	55
5.21.2.2	bb_bos_dscr_offset	56
5.21.2.3	bb_bus_power	56
5.21.2.4	bb_ec_present	56
5.21.2.5	bb_enable	56
5.21.2.6	bb_option	56
5.21.2.7	bb_pid	56
5.21.2.8	bb_string_dscr_offset	56
5.21.2.9	bb_timeout	56
5.21.2.10	bb_unique_container_id	57
5.21.2.11	bb_vid	57
5.21.2.12	reserved_1	57
5.21.2.13	table_len	57
5.22	bc_dp_dm_state_t Union Reference	57
5.22.1	Detailed Description	57
5.22.2	Field Documentation	57
5.22.2.1	d	57

5.22.2.2	state	58
5.23	bc_status_t Struct Reference	58
5.23.1	Detailed Description	58
5.23.2	Field Documentation	58
5.23.2.1	afc_src_cur_match_byte	58
5.23.2.2	afc_src_is_matched	58
5.23.2.3	afc_src_last_match_byte	59
5.23.2.4	afc_src_match_count	59
5.23.2.5	afc_src_matched_byte	59
5.23.2.6	afc_src_msg_count	59
5.23.2.7	afc_tx_active	59
5.23.2.8	attach	59
5.23.2.9	bc_evt	59
5.23.2.10	bc_fsm_state	59
5.23.2.11	comp_rising	59
5.23.2.12	connected	60
5.23.2.13	cur_amp	60
5.23.2.14	cur_mode	60
5.23.2.15	cur_timer	60
5.23.2.16	cur_volt	60
5.23.2.17	dp_dm_status	60
5.23.2.18	old_dp_dm_status	60
5.24	pd_do_t::BIST_DO Struct Reference	60
5.24.1	Detailed Description	61
5.24.2	Field Documentation	61
5.24.2.1	mode	61
5.24.2.2	rsvd1	61
5.24.2.3	rsvd2	61
5.25	cc_state_t Union Reference	61
5.25.1	Detailed Description	61
5.25.2	Field Documentation	61
5.25.2.1	cc	61
5.25.2.2	state	62
5.26	ccg_timer_t Struct Reference	62
5.26.1	Detailed Description	62
5.26.2	Field Documentation	62
5.26.2.1	cb	62
5.26.2.2	count	62
5.26.2.3	id	62
5.26.2.4	period	63

5.27	chg_cfg_params_t Struct Reference	63
5.27.1	Detailed Description	63
5.27.2	Field Documentation	63
5.27.2.1	afc_src_cap_cnt	63
5.27.2.2	afc_src_caps	63
5.27.2.3	apple_src_id	63
5.27.2.4	qc_src_type	64
5.27.2.5	reserved_0	64
5.27.2.6	reserved_1	64
5.27.2.7	snk_sel	64
5.27.2.8	src_sel	64
5.27.2.9	table_len	64
5.28	contract_t Struct Reference	64
5.28.1	Detailed Description	64
5.28.2	Field Documentation	65
5.28.2.1	cur_pwr	65
5.28.2.2	max_volt	65
5.28.2.3	min_volt	65
5.29	custom_alt_cfg_settings_t Struct Reference	65
5.29.1	Detailed Description	65
5.29.2	Field Documentation	65
5.29.2.1	custom_alt_mode	65
5.29.2.2	custom_dfp_mask	66
5.29.2.3	custom_ufp_mask	66
5.29.2.4	reserved0	66
5.29.2.5	reserved1	66
5.29.2.6	table_len	66
5.30	custom_host_cfg_settings_t Struct Reference	66
5.30.1	Detailed Description	66
5.30.2	Field Documentation	67
5.30.2.1	acc_mode_disable	67
5.30.2.2	ext_powered_prs	67
5.30.2.3	pdo_sel_alg	67
5.30.2.4	pwr_threshold	67
5.30.2.5	req_max_pwr	67
5.30.2.6	reserved	67
5.30.2.7	rp_detach_disable	67
5.30.2.8	snk_path_enable	67
5.30.2.9	table_len	68
5.31	pd_do_t::DFP_VDO Struct Reference	68

5.31.1	Detailed Description	68
5.31.2	Field Documentation	68
5.31.2.1	host_cap	68
5.31.2.2	port_num	68
5.31.2.3	rsvd0	68
5.31.2.4	rsvd1	68
5.31.2.5	vdo_version	69
5.32	pd_do_t::DP_CONFIG_VDO Struct Reference	69
5.32.1	Detailed Description	69
5.32.2	Field Documentation	69
5.32.2.1	dfp_asgmt	69
5.32.2.2	rsvd1	69
5.32.2.3	rsvd2	69
5.32.2.4	sel_conf	69
5.32.2.5	sign	70
5.32.2.6	ufp_asgmt	70
5.33	pd_do_t::DP_STATUS_VDO Struct Reference	70
5.33.1	Detailed Description	70
5.33.2	Field Documentation	70
5.33.2.1	dfp_ufp_conn	70
5.33.2.2	en	70
5.33.2.3	exit	71
5.33.2.4	hpd_irq	71
5.33.2.5	hpd_state	71
5.33.2.6	mult_fun	71
5.33.2.7	pwr_low	71
5.33.2.8	rsvd	71
5.33.2.9	usb_cfg	71
5.34	dpm_pd_cmd_buf_t Struct Reference	71
5.34.1	Detailed Description	72
5.34.2	Field Documentation	72
5.34.2.1	cmd_do	72
5.34.2.2	cmd_sop	72
5.34.2.3	dat_ptr	72
5.34.2.4	extd_hdr	72
5.34.2.5	extd_type	72
5.34.2.6	no_of_cmd_do	72
5.34.2.7	timeout	73
5.35	dpm_status_t Struct Reference	73
5.35.1	Detailed Description	75

5.35.2	Field Documentation	75
5.35.2.1	alert	75
5.35.2.2	app_cbk	76
5.35.2.3	attach	76
5.35.2.4	attached_dev	76
5.35.2.5	bist_cm2_enabled	76
5.35.2.6	bootup	76
5.35.2.7	cbl_dsc	76
5.35.2.8	cbl_mode_en	76
5.35.2.9	cbl_soft_reset_tried	76
5.35.2.10	cbl_state	76
5.35.2.11	cbl_type	77
5.35.2.12	cbl_vdm_version	77
5.35.2.13	cbl_vdo	77
5.35.2.14	cbl_vdo_2	77
5.35.2.15	cbl_wait	77
5.35.2.16	cc_live	77
5.35.2.17	cc_old_status	77
5.35.2.18	cc_rd_status	77
5.35.2.19	cc_status	77
5.35.2.20	cmd_p	78
5.35.2.21	connect	78
5.35.2.22	contract	78
5.35.2.23	contract_exist	78
5.35.2.24	cur_fb	78
5.35.2.25	cur_port_role	78
5.35.2.26	cur_port_type	78
5.35.2.27	cur_snk_max_min	78
5.35.2.28	cur_snk_pdo	78
5.35.2.29	cur_snk_pdo_count	79
5.35.2.30	cur_src_pdo	79
5.35.2.31	cur_src_pdo_count	79
5.35.2.32	db_support	79
5.35.2.33	dead_bat	79
5.35.2.34	dflt_port_role	79
5.35.2.35	dpm_cmd_buf	79
5.35.2.36	dpm_enabled	79
5.35.2.37	dpm_err_info	79
5.35.2.38	dpm_init	80
5.35.2.39	dpm_pd_cbk	80

5.35.2.40 dpm_pd_cmd	80
5.35.2.41 dpm_pd_cmd_active	80
5.35.2.42 dpm_safe_disable	80
5.35.2.43 dpm_typec_cbk	80
5.35.2.44 dpm_typec_cmd	80
5.35.2.45 dpm_typec_cmd_active	80
5.35.2.46 drp_period	80
5.35.2.47 emca_present	81
5.35.2.48 err_recov	81
5.35.2.49 ext_snk_cap	81
5.35.2.50 ext_src_cap	81
5.35.2.51 fault_active	81
5.35.2.52 fr_rx_disabled	81
5.35.2.53 fr_rx_en_live	81
5.35.2.54 fr_tx_disabled	81
5.35.2.55 fr_tx_en_live	81
5.35.2.56 frs_enable	82
5.35.2.57 frs_rx_en	82
5.35.2.58 frs_tx_en	82
5.35.2.59 hw_drp_toggle_en	82
5.35.2.60 is_snk_bat	82
5.35.2.61 is_src_bat	82
5.35.2.62 non_intr_response	82
5.35.2.63 padval	82
5.35.2.64 pd3_src_cc_busy	82
5.35.2.65 pd_connected	83
5.35.2.66 pd_disabled	83
5.35.2.67 pd_support	83
5.35.2.68 pe_evt	83
5.35.2.69 pe_fsm_state	83
5.35.2.70 polarity	83
5.35.2.71 port_disable	83
5.35.2.72 port_role	83
5.35.2.73 port_status	83
5.35.2.74 pps_src_en	84
5.35.2.75 pps_status	84
5.35.2.76 pwr_limited	84
5.35.2.77 ra_present	84
5.35.2.78 rand_base	84
5.35.2.79 reserved_3	84

5.35.2.80 rev3_en	84
5.35.2.81 rev_pol	84
5.35.2.82 role_at_connect	84
5.35.2.83 rp_supported	85
5.35.2.84 skip_scan	85
5.35.2.85 snk_cur_level	85
5.35.2.86 snk_max_min	85
5.35.2.87 snk_pdo	85
5.35.2.88 snk_pdo_count	85
5.35.2.89 snk_pdo_mask	85
5.35.2.90 snk_period	85
5.35.2.91 snk_rdo	86
5.35.2.92 snk_rp_detach_en	86
5.35.2.93 snk_sel_pdo	86
5.35.2.94 snk_usb_comm_en	86
5.35.2.95 snk_usb_susp_en	86
5.35.2.96 spec_rev_cbl	86
5.35.2.97 spec_rev_peer	86
5.35.2.98 spec_rev_sop_live	86
5.35.2.99 spec_rev_sop_prime_live	86
5.35.2.100src_cap_p	87
5.35.2.101src_cap_start_delay	87
5.35.2.102src_cur_level	87
5.35.2.103src_cur_level_live	87
5.35.2.104src_cur_rdo	87
5.35.2.105src_last_rdo	87
5.35.2.106src_pdo	87
5.35.2.107src_pdo_count	87
5.35.2.108src_pdo_mask	87
5.35.2.109src_period	88
5.35.2.110src_rdo	88
5.35.2.111src_sel_pdo	88
5.35.2.112swap_response	88
5.35.2.113toggle	88
5.35.2.114try_src_snk	88
5.35.2.115try_src_snk_dis	88
5.35.2.116typec_evt	88
5.35.2.117typec_fsm_state	89
5.35.2.118unchunk_sup_live	89
5.35.2.119unchunk_sup_peer	89

5.35.2.120usb4_en	89
5.35.2.121vconn_logical	89
5.35.2.122vconn_retain	89
5.36 pd_do_t::ENTERUSB_VDO Struct Reference	89
5.36.1 Detailed Description	90
5.36.2 Field Documentation	90
5.36.2.1 cable_current	90
5.36.2.2 cable_speed	90
5.36.2.3 cable_type	90
5.36.2.4 host_dp_supp	90
5.36.2.5 host_pcie_supp	90
5.36.2.6 host_present	90
5.36.2.7 host_tbt_supp	90
5.36.2.8 rsvd0	91
5.36.2.9 rsvd1	91
5.36.2.10 rsvd2	91
5.36.2.11 rsvd3	91
5.36.2.12 usb3_drd	91
5.36.2.13 usb4_drd	91
5.36.2.14 usb_mode	91
5.37 pd_extd_hdr_t::EXTD_HDR_T Struct Reference	91
5.37.1 Detailed Description	92
5.37.2 Field Documentation	92
5.37.2.1 chunk_no	92
5.37.2.2 chunked	92
5.37.2.3 data_size	92
5.37.2.4 request	92
5.37.2.5 rsvd1	92
5.38 pd_do_t::FIXED_SNK Struct Reference	92
5.38.1 Detailed Description	93
5.38.2 Field Documentation	93
5.38.2.1 dr_swap	93
5.38.2.2 dual_role_power	93
5.38.2.3 ext_powered	93
5.38.2.4 fr_swap	93
5.38.2.5 high_cap	93
5.38.2.6 op_current	93
5.38.2.7 rsrvd	94
5.38.2.8 supply_type	94
5.38.2.9 usb_comm_cap	94

5.38.2.10 voltage	94
5.39 pd_do_t::FIXED_SRC Struct Reference	94
5.39.1 Detailed Description	94
5.39.2 Field Documentation	94
5.39.2.1 dr_swap	95
5.39.2.2 dual_role_power	95
5.39.2.3 ext_powered	95
5.39.2.4 max_current	95
5.39.2.5 pk_current	95
5.39.2.6 reserved	95
5.39.2.7 supply_type	95
5.39.2.8 unchunk_sup	95
5.39.2.9 usb_comm_cap	95
5.39.2.10 usb_suspend_sup	96
5.39.2.11 voltage	96
5.40 fw_img_status_t Union Reference	96
5.40.1 Detailed Description	96
5.40.2 Field Documentation	96
5.40.2.1 status	96
5.40.2.2 val	96
5.41 fw_img_status_t::fw_mode_reason_t Struct Reference	97
5.41.1 Detailed Description	97
5.41.2 Field Documentation	97
5.41.2.1 boot_mode_request	97
5.41.2.2 fw1_invalid	97
5.41.2.3 fw2_invalid	97
5.41.2.4 reserved	97
5.41.2.5 reserved1	97
5.42 i2c_scb_config_t Struct Reference	98
5.42.1 Detailed Description	98
5.42.2 Field Documentation	98
5.42.2.1 buf_size	98
5.42.2.2 buffer	98
5.42.2.3 cb_fun_ptr	98
5.42.2.4 clock_freq	98
5.42.2.5 i2c_state	99
5.42.2.6 i2c_write_count	99
5.42.2.7 mode	99
5.42.2.8 slave_address	99
5.42.2.9 slave_mask	99

5.43	icl_tgl_cfg_settings_t Struct Reference	99
5.43.1	Detailed Description	99
5.43.2	Field Documentation	100
5.43.2.1	icl_dual_retimer_enable	100
5.43.2.2	icl_i2c_pmc_address	100
5.43.2.3	icl_i2c_retimer_address	100
5.43.2.4	icl_tgl_selection	100
5.43.2.5	reserved0	100
5.43.2.6	soc_mux_config_delay	100
5.43.2.7	soc_mux_init_delay	100
5.43.2.8	table_len	100
5.44	ocp_settings_t Struct Reference	101
5.44.1	Detailed Description	101
5.44.2	Field Documentation	101
5.44.2.1	cs_res	101
5.44.2.2	debounce	101
5.44.2.3	debounce2	101
5.44.2.4	retry_cnt	101
5.44.2.5	sense_res	101
5.44.2.6	table_len	102
5.44.2.7	threshold	102
5.44.2.8	threshold2	102
5.45	otp_settings_t Struct Reference	102
5.45.1	Detailed Description	102
5.45.2	Field Documentation	102
5.45.2.1	cutoff_val	102
5.45.2.2	cutoff_val_1	103
5.45.2.3	debounce	103
5.45.2.4	reserved_1	103
5.45.2.5	restart_val	103
5.45.2.6	restart_val_1	103
5.45.2.7	table_len	103
5.45.2.8	therm_1_enable	103
5.45.2.9	therm_type	103
5.45.2.10	therm_type_1	103
5.46	ovp_settings_t Struct Reference	104
5.46.1	Detailed Description	104
5.46.2	Field Documentation	104
5.46.2.1	debounce	104
5.46.2.2	debounce_ms	104

5.46.2.3	retry_cnt	104
5.46.2.4	table_len	104
5.46.2.5	threshold	104
5.47	pd_do_t::PAS_CBL_VDO Struct Reference	105
5.47.1	Detailed Description	105
5.47.2	Field Documentation	105
5.47.2.1	cbl_fw_ver	105
5.47.2.2	cbl_hw_ver	105
5.47.2.3	cbl_latency	105
5.47.2.4	cbl_term	106
5.47.2.5	max_vbus_volt	106
5.47.2.6	rsvd1	106
5.47.2.7	rsvd2	106
5.47.2.8	rsvd3	106
5.47.2.9	typec_abc	106
5.47.2.10	typec_plug	106
5.47.2.11	usb_ss_sup	106
5.47.2.12	vbus_cur	106
5.47.2.13	vdo_version	107
5.48	pasc_valley_table_t Struct Reference	107
5.48.1	Detailed Description	107
5.48.2	Field Documentation	107
5.48.2.1	max_cur	107
5.48.2.2	reserved_0	107
5.48.2.3	safe_valley	107
5.48.2.4	table	108
5.48.2.5	v0	108
5.48.2.6	v1	108
5.48.2.7	v2	108
5.49	pd_config_t Struct Reference	108
5.49.1	Detailed Description	109
5.49.2	Field Documentation	109
5.49.2.1	application	109
5.49.2.2	cable_disc_cnt	109
5.49.2.3	db_event_mask	109
5.49.2.4	flash_checksum	109
5.49.2.5	flashing_mode	109
5.49.2.6	flashing_vid	109
5.49.2.7	mfg_info_length	109
5.49.2.8	mfg_info_offset	109

5.49.2.9	pd_port_cnt	110
5.49.2.10	port_conf	110
5.49.2.11	reserved_0	110
5.49.2.12	reserved_2	110
5.49.2.13	table_checksum	110
5.49.2.14	table_sign	110
5.49.2.15	table_size	110
5.49.2.16	table_type	110
5.49.2.17	table_version	110
5.50	pd_contract_info_t Struct Reference	111
5.50.1	Detailed Description	111
5.50.2	Field Documentation	111
5.50.2.1	rdo	111
5.50.2.2	status	111
5.51	pd_do_t Union Reference	111
5.51.1	Detailed Description	113
5.51.2	Field Documentation	113
5.51.2.1	act_cbl_vdo	113
5.51.2.2	act_cbl_vdo1	113
5.51.2.3	act_cbl_vdo2	113
5.51.2.4	ado_alert	114
5.51.2.5	bat_snk	114
5.51.2.6	bat_src	114
5.51.2.7	bist_do	114
5.51.2.8	dfp_vdo	114
5.51.2.9	dp_cfg_vdo	114
5.51.2.10	dp_stat_vdo	114
5.51.2.11	enterusb_vdo	114
5.51.2.12	fixed_snk	114
5.51.2.13	fixed_src	115
5.51.2.14	pas_cbl_vdo	115
5.51.2.15	pps_snk	115
5.51.2.16	pps_src	115
5.51.2.17	qc_4_0_data_vdo	115
5.51.2.18	rdo_bat	115
5.51.2.19	rdo_bat_gvb	115
5.51.2.20	rdo_fix_var	115
5.51.2.21	rdo_fix_var_gvb	115
5.51.2.22	rdo_gen	116
5.51.2.23	rdo_gen_gvb	116

5.51.2.24 rdo_pps	116
5.51.2.25 src_gen	116
5.51.2.26 std_ama_vdo	116
5.51.2.27 std_ama_vdo_pd3	116
5.51.2.28 std_cbl_vdo	116
5.51.2.29 std_cert_vdo	116
5.51.2.30 std_dp_vdo	116
5.51.2.31 std_id_hdr	117
5.51.2.32 std_prod_vdo	117
5.51.2.33 std_svid_res	117
5.51.2.34 std_vdm_hdr	117
5.51.2.35 tbt_cbl_vdo	117
5.51.2.36 tbt_ufp_vdo	117
5.51.2.37 tbt_vdo	117
5.51.2.38 ufp_vdo_1	117
5.51.2.39 ustd_qc_4_0_hdr	117
5.51.2.40 ustd_vdm_hdr	118
5.51.2.41 val	118
5.51.2.42 var_snk	118
5.51.2.43 var_src	118
5.52 pd_extd_hdr_t Union Reference	118
5.52.1 Detailed Description	118
5.52.2 Field Documentation	118
5.52.2.1 extd	118
5.52.2.2 val	119
5.53 pd_hdr_t::PD_HDR Struct Reference	119
5.53.1 Detailed Description	119
5.53.2 Field Documentation	119
5.53.2.1 chunk_no	119
5.53.2.2 chunked	119
5.53.2.3 data_role	119
5.53.2.4 data_size	120
5.53.2.5 extd	120
5.53.2.6 len	120
5.53.2.7 msg_id	120
5.53.2.8 msg_type	120
5.53.2.9 pwr_role	120
5.53.2.10 request	120
5.53.2.11 rsvd1	120
5.53.2.12 spec_rev	120

5.54	pd_hdr_t Union Reference	121
5.54.1	Detailed Description	121
5.54.2	Field Documentation	121
5.54.2.1	hdr	121
5.54.2.2	val	121
5.55	pd_packet_extd_t Struct Reference	121
5.55.1	Detailed Description	122
5.55.2	Field Documentation	122
5.55.2.1	dat	122
5.55.2.2	data_role	122
5.55.2.3	hdr	122
5.55.2.4	len	122
5.55.2.5	msg	122
5.55.2.6	sop	122
5.56	pd_packet_t Struct Reference	122
5.56.1	Detailed Description	123
5.56.2	Field Documentation	123
5.56.2.1	dat	123
5.56.2.2	data_role	123
5.56.2.3	hdr	123
5.56.2.4	len	123
5.56.2.5	msg	123
5.56.2.6	sop	123
5.57	pd_port_config_t Struct Reference	124
5.57.1	Detailed Description	125
5.57.2	Field Documentation	125
5.57.2.1	alt_mode_tbl_offset	125
5.57.2.2	alt_mode_trigger	125
5.57.2.3	auto_cfg_tbl_offset	125
5.57.2.4	bat_chg_tbl_offset	126
5.57.2.5	bb_tbl_offset	126
5.57.2.6	cable_disc_en	126
5.57.2.7	chg_cfg_tbl_offset	126
5.57.2.8	current_level	126
5.57.2.9	custom_alt_mode_tbl_offset	126
5.57.2.10	custom_host_tbl_offset	126
5.57.2.11	dead_bat_support	126
5.57.2.12	default_role	126
5.57.2.13	default_sink_pdo_mask	127
5.57.2.14	default_src_pdo_mask	127

5.57.2.15 dock_cfg_tbl_offset	127
5.57.2.16 dp_config_supported	127
5.57.2.17 dp_mux_control	127
5.57.2.18 dp_oper	127
5.57.2.19 dp_pref_mode	127
5.57.2.20 drp_toggle_en	127
5.57.2.21 err_recovery_en	127
5.57.2.22 ext_snk_cap_length	128
5.57.2.23 ext_snk_cap_offset	128
5.57.2.24 ext_src_cap_length	128
5.57.2.25 ext_src_cap_offset	128
5.57.2.26 frs_enable	128
5.57.2.27 icl_tgl_tbl_offset	128
5.57.2.28 id_vdm_length	128
5.57.2.29 id_vdm_offset	128
5.57.2.30 is_snk_bat	128
5.57.2.31 is_src_bat	129
5.57.2.32 mode_vdm_length	129
5.57.2.33 mode_vdm_offset	129
5.57.2.34 ocp_tbl_offset	129
5.57.2.35 otp_tbl_offset	129
5.57.2.36 ovp_tbl_offset	129
5.57.2.37 pd_operation_en	129
5.57.2.38 port_disable	129
5.57.2.39 port_role	129
5.57.2.40 protection_enable	130
5.57.2.41 pwr_tbl_offset	130
5.57.2.42 ra_timeout	130
5.57.2.43 rcp_tbl_offset	130
5.57.2.44 reserved_0	130
5.57.2.45 reserved_1	130
5.57.2.46 reserved_10	130
5.57.2.47 reserved_11	130
5.57.2.48 reserved_3	130
5.57.2.49 reserved_4	131
5.57.2.50 reserved_6	131
5.57.2.51 reserved_8	131
5.57.2.52 reserved_9	131
5.57.2.53 rp_supported	131
5.57.2.54 scp_tbl_offset	131

5.57.2.55 snk_pdo_cnt	131
5.57.2.56 snk_pdo_list	131
5.57.2.57 snk_pdo_max_min_current_pwr	131
5.57.2.58 snk_usb_comm_en	132
5.57.2.59 snk_usb_susp_en	132
5.57.2.60 spm_cfg_tbl_offset	132
5.57.2.61 src_pdo_cnt	132
5.57.2.62 src_pdo_list	132
5.57.2.63 svid_vdm_length	132
5.57.2.64 svid_vdm_offset	132
5.57.2.65 swap_response	132
5.57.2.66 tbthost_cfg_tbl_offset	132
5.57.2.67 try_src_en	133
5.57.2.68 type_a_chg_tbl_offset	133
5.57.2.69 type_a_enable	133
5.57.2.70 type_a_pwr_tbl_offset	133
5.57.2.71 uvp_tbl_offset	133
5.57.2.72 vconn_ocp_tbl_offset	133
5.57.2.73 vconn_retain	133
5.58 pd_port_status_t::PD_PORT_STAT Struct Reference	133
5.58.1 Detailed Description	134
5.58.2 Field Documentation	134
5.58.2.1 ccg_spec_rev	134
5.58.2.2 contract_exist	134
5.58.2.3 cur_data_role	134
5.58.2.4 cur_power_role	134
5.58.2.5 dflt_data_pref	134
5.58.2.6 dflt_data_role	135
5.58.2.7 dflt_power_pref	135
5.58.2.8 dflt_power_role	135
5.58.2.9 emca_present	135
5.58.2.10 emca_spec_rev	135
5.58.2.11 emca_type	135
5.58.2.12 min_state	135
5.58.2.13 pe_rdy	135
5.58.2.14 peer_pd3_supp	135
5.58.2.15 peer_unchunk_supp	136
5.58.2.16 reserved0	136
5.58.2.17 reserved2	136
5.58.2.18 rp_status	136

5.58.2.19 vconn_on	136
5.58.2.20 vconn_src	136
5.59 pd_port_status_t Union Reference	136
5.59.1 Detailed Description	137
5.59.2 Field Documentation	137
5.59.2.1 status	137
5.59.2.2 val	137
5.60 pd_power_status_t Struct Reference	137
5.60.1 Detailed Description	137
5.60.2 Field Documentation	137
5.60.2.1 battery_input	137
5.60.2.2 dummy	138
5.60.2.3 event_flags	138
5.60.2.4 intl_temperature	138
5.60.2.5 power_status	138
5.60.2.6 present_input	138
5.60.2.7 temp_status	138
5.61 pd_stack_conf_t Struct Reference	138
5.61.1 Detailed Description	138
5.61.2 Field Documentation	139
5.61.2.1 frs_rx	139
5.61.2.2 frs_tx	139
5.61.2.3 pd_rev_3	139
5.61.2.4 source_only	139
5.62 pd_status_t Struct Reference	139
5.62.1 Detailed Description	140
5.62.2 Field Documentation	140
5.62.2.1 avoid_retry	140
5.62.2.2 bist_test_en	140
5.62.2.3 cbk	140
5.62.2.4 ctrs	141
5.62.2.5 cur_rec_msg_id	141
5.62.2.6 frs_rx_enable	141
5.62.2.7 frs_tx_enable	141
5.62.2.8 last_rcvd_sop	141
5.62.2.9 last_tx_sop	141
5.62.2.10 rev3_enable	141
5.62.2.11 rx_busy	141
5.62.2.12 rx_evt	141
5.62.2.13 rx_packet	142

5.62.2.14 tx_buf	142
5.62.2.15 tx_busy	142
5.62.2.16 tx_count	142
5.62.2.17 tx_extd	142
5.62.2.18 tx_extd_hdr	142
5.62.2.19 tx_header	142
5.62.2.20 tx_msg_type	142
5.62.2.21 tx_sop	142
5.63 pd_do_t::PPS_SNK Struct Reference	143
5.63.1 Detailed Description	143
5.63.2 Field Documentation	143
5.63.2.1 apdo_type	143
5.63.2.2 cur_fb	143
5.63.2.3 max_volt	143
5.63.2.4 min_volt	143
5.63.2.5 op_cur	143
5.63.2.6 rsvd1	144
5.63.2.7 rsvd2	144
5.63.2.8 rsvd3	144
5.63.2.9 rsvd4	144
5.63.2.10 supply_type	144
5.64 pd_do_t::PPS_SRC Struct Reference	144
5.64.1 Detailed Description	144
5.64.2 Field Documentation	145
5.64.2.1 apdo_type	145
5.64.2.2 max_cur	145
5.64.2.3 max_volt	145
5.64.2.4 min_volt	145
5.64.2.5 pps_pwr_limited	145
5.64.2.6 rsvd1	145
5.64.2.7 rsvd2	145
5.64.2.8 rsvd3	145
5.64.2.9 supply_type	146
5.65 prl_cntrs_t Struct Reference	146
5.65.1 Detailed Description	146
5.65.2 Field Documentation	146
5.65.2.1 first_msg_rcvd	146
5.65.2.2 rec_msg_id	146
5.65.2.3 tr_msg_id	146
5.66 pwr_params_t Struct Reference	146

5.66.1	Detailed Description	147
5.66.2	Field Documentation	147
5.66.2.1	cable_resistance	147
5.66.2.2	cur_sense_res	147
5.66.2.3	fb_ctrl_r1	147
5.66.2.4	fb_ctrl_r2	148
5.66.2.5	fb_type	148
5.66.2.6	max_pwm_duty_cycle	148
5.66.2.7	prim_sec_turns_ratio	148
5.66.2.8	pwm_fix_freq	148
5.66.2.9	pwm_max_freq	148
5.66.2.10	pwm_min_freq	148
5.66.2.11	pwm_mode	148
5.66.2.12	reserved_0	148
5.66.2.13	reserved_2	149
5.66.2.14	reserved_3	149
5.66.2.15	sr_async_thresh	149
5.66.2.16	sr_enable	149
5.66.2.17	sr_fall_time	149
5.66.2.18	sr_rise_time	149
5.66.2.19	sr_supply_doubler	149
5.66.2.20	src_gate_drv_str	149
5.66.2.21	table_len	149
5.66.2.22	vbtr_down_step_width	150
5.66.2.23	vbtr_up_step_width	150
5.66.2.24	vbus_dflt_volt	150
5.66.2.25	vbus_max_volt	150
5.66.2.26	vbus_min_volt	150
5.66.2.27	vbus_offset_volt	150
5.67	pd_do_t::QC_4_0_DATA_VDO Struct Reference	150
5.67.1	Detailed Description	150
5.67.2	Field Documentation	151
5.67.2.1	data_0	151
5.67.2.2	data_1	151
5.67.2.3	data_2	151
5.67.2.4	data_3	151
5.68	rcp_settings_t Struct Reference	151
5.68.1	Detailed Description	151
5.68.2	Field Documentation	151
5.68.2.1	resvd	152

5.68.2.2	retry_cnt	152
5.68.2.3	table_len	152
5.69	pd_do_t::RDO_BAT Struct Reference	152
5.69.1	Detailed Description	152
5.69.2	Field Documentation	152
5.69.2.1	cap_mismatch	152
5.69.2.2	give_back_flag	153
5.69.2.3	max_op_power	153
5.69.2.4	no_usb_suspend	153
5.69.2.5	obj_pos	153
5.69.2.6	op_power	153
5.69.2.7	rsrvd1	153
5.69.2.8	rsrvd2	153
5.69.2.9	unchunk_sup	153
5.69.2.10	usb_comm_cap	153
5.70	pd_do_t::RDO_BAT_GIVEBACK Struct Reference	154
5.70.1	Detailed Description	154
5.70.2	Field Documentation	154
5.70.2.1	cap_mismatch	154
5.70.2.2	give_back_flag	154
5.70.2.3	min_op_power	154
5.70.2.4	no_usb_suspend	154
5.70.2.5	obj_pos	154
5.70.2.6	op_power	155
5.70.2.7	rsrvd1	155
5.70.2.8	rsrvd2	155
5.70.2.9	unchunk_sup	155
5.70.2.10	usb_comm_cap	155
5.71	pd_do_t::RDO_FIXED_VAR Struct Reference	155
5.71.1	Detailed Description	155
5.71.2	Field Documentation	156
5.71.2.1	cap_mismatch	156
5.71.2.2	give_back_flag	156
5.71.2.3	max_op_current	156
5.71.2.4	no_usb_suspend	156
5.71.2.5	obj_pos	156
5.71.2.6	op_current	156
5.71.2.7	rsrvd1	156
5.71.2.8	rsrvd2	156
5.71.2.9	unchunk_sup	157

5.71.2.10	usb_comm_cap	157
5.72	pd_do_t::RDO_FIXED_VAR_GIVEBACK Struct Reference	157
5.72.1	Detailed Description	157
5.72.2	Field Documentation	157
5.72.2.1	cap_mismatch	157
5.72.2.2	give_back_flag	157
5.72.2.3	min_op_current	158
5.72.2.4	no_usb_suspend	158
5.72.2.5	obj_pos	158
5.72.2.6	op_current	158
5.72.2.7	rsrvd1	158
5.72.2.8	rsrvd2	158
5.72.2.9	unchunk_sup	158
5.72.2.10	usb_comm_cap	158
5.73	pd_do_t::RDO_GEN Struct Reference	158
5.73.1	Detailed Description	159
5.73.2	Field Documentation	159
5.73.2.1	cap_mismatch	159
5.73.2.2	give_back_flag	159
5.73.2.3	min_max_power_cur	159
5.73.2.4	no_usb_suspend	159
5.73.2.5	obj_pos	159
5.73.2.6	op_power_cur	160
5.73.2.7	rsrvd1	160
5.73.2.8	rsrvd2	160
5.73.2.9	unchunk_sup	160
5.73.2.10	usb_comm_cap	160
5.74	pd_do_t::RDO_GEN_GVB Struct Reference	160
5.74.1	Detailed Description	160
5.74.2	Field Documentation	161
5.74.2.1	cap_mismatch	161
5.74.2.2	give_back_flag	161
5.74.2.3	max_power_cur	161
5.74.2.4	no_usb_suspend	161
5.74.2.5	obj_pos	161
5.74.2.6	op_power_cur	161
5.74.2.7	rsrvd1	161
5.74.2.8	rsrvd2	161
5.74.2.9	unchunk_sup	162
5.74.2.10	usb_comm_cap	162

5.75	pd_do_t::RDO_PPS Struct Reference	162
5.75.1	Detailed Description	162
5.75.2	Field Documentation	162
5.75.2.1	cap_mismatch	162
5.75.2.2	no_usb_suspend	162
5.75.2.3	obj_pos	163
5.75.2.4	op_cur	163
5.75.2.5	out_volt	163
5.75.2.6	rsvd1	163
5.75.2.7	rsvd2	163
5.75.2.8	rsvd3	163
5.75.2.9	rsvd4	163
5.75.2.10	unchunk_sup	163
5.75.2.11	usb_comm_cap	163
5.76	reg_am_t Struct Reference	164
5.76.1	Detailed Description	164
5.76.2	Field Documentation	164
5.76.2.1	reg_am_ptr	164
5.76.2.2	svid	164
5.77	scp_settings_t Struct Reference	164
5.77.1	Detailed Description	164
5.77.2	Field Documentation	164
5.77.2.1	debounce	165
5.77.2.2	retry_cnt	165
5.77.2.3	table_len	165
5.77.2.4	threshold	165
5.78	sensor_data_t Struct Reference	165
5.78.1	Detailed Description	165
5.78.2	Field Documentation	165
5.78.2.1	sensor_ctrl	165
5.78.2.2	sensor_oc1	166
5.78.2.3	sensor_oc2	166
5.78.2.4	sensor_oc3	166
5.79	pd_do_t::SRC_GEN Struct Reference	166
5.79.1	Detailed Description	166
5.79.2	Field Documentation	166
5.79.2.1	max_cur_power	166
5.79.2.2	max_voltage	166
5.79.2.3	min_voltage	167
5.79.2.4	supply_type	167

5.80	pd_do_t::STD_AMA_VDO Struct Reference	167
5.80.1	Detailed Description	167
5.80.2	Field Documentation	167
5.80.2.1	ama_fw_ver	167
5.80.2.2	ama_hw_ver	167
5.80.2.3	rsvd1	168
5.80.2.4	ssrx1	168
5.80.2.5	ssrx2	168
5.80.2.6	sstx1	168
5.80.2.7	sstx2	168
5.80.2.8	usb_ss_sup	168
5.80.2.9	vbus_req	168
5.80.2.10	vcon_pwr	168
5.80.2.11	vcon_req	168
5.81	pd_do_t::STD_AMA_VDO_PD3 Struct Reference	169
5.81.1	Detailed Description	169
5.81.2	Field Documentation	169
5.81.2.1	ama_fw_ver	169
5.81.2.2	ama_hw_ver	169
5.81.2.3	rsvd1	169
5.81.2.4	usb_ss_sup	169
5.81.2.5	vbus_req	169
5.81.2.6	vcon_pwr	170
5.81.2.7	vcon_req	170
5.81.2.8	vdo_version	170
5.82	pd_do_t::STD_CBL_VDO Struct Reference	170
5.82.1	Detailed Description	170
5.82.2	Field Documentation	170
5.82.2.1	cbl_fw_ver	171
5.82.2.2	cbl_hw_ver	171
5.82.2.3	cbl_latency	171
5.82.2.4	cbl_term	171
5.82.2.5	rsvd1	171
5.82.2.6	sop_dp	171
5.82.2.7	ssrx1	171
5.82.2.8	ssrx2	171
5.82.2.9	sstx1	171
5.82.2.10	sstx2	172
5.82.2.11	typec_abc	172
5.82.2.12	typec_plug	172

5.82.2.13	usb_ss_sup	172
5.82.2.14	vbus_cur	172
5.82.2.15	vbus_thru_cbl	172
5.83	pd_do_t::STD_CERT_VDO Struct Reference	172
5.83.1	Detailed Description	172
5.83.2	Field Documentation	173
5.83.2.1	usb_xid	173
5.84	pd_do_t::STD_DP_VDO Struct Reference	173
5.84.1	Detailed Description	173
5.84.2	Field Documentation	173
5.84.2.1	dfp_d_pin	173
5.84.2.2	port_cap	173
5.84.2.3	recep	173
5.84.2.4	rsvd	174
5.84.2.5	signal	174
5.84.2.6	ufp_d_pin	174
5.84.2.7	usb2_0	174
5.85	pd_do_t::STD_PROD_VDO Struct Reference	174
5.85.1	Detailed Description	174
5.85.2	Field Documentation	174
5.85.2.1	bcd_dev	174
5.85.2.2	usb_pid	175
5.86	pd_do_t::STD_SVID_RESP_VDO Struct Reference	175
5.86.1	Detailed Description	175
5.86.2	Field Documentation	175
5.86.2.1	svid_n	175
5.86.2.2	svid_n1	175
5.87	pd_do_t::STD_VDM_HDR Struct Reference	175
5.87.1	Detailed Description	176
5.87.2	Field Documentation	176
5.87.2.1	cmd	176
5.87.2.2	cmd_type	176
5.87.2.3	obj_pos	176
5.87.2.4	rsvd1	176
5.87.2.5	rsvd2	176
5.87.2.6	st_ver	176
5.87.2.7	svid	176
5.87.2.8	vdm_type	177
5.88	pd_do_t::STD_VDM_ID_HDR Struct Reference	177
5.88.1	Detailed Description	177

5.88.2	Field Documentation	177
5.88.2.1	mod_support	177
5.88.2.2	prod_type	177
5.88.2.3	prod_type_dfp	177
5.88.2.4	rsvd1	178
5.88.2.5	usb_dev	178
5.88.2.6	usb_host	178
5.88.2.7	usb_vid	178
5.89	sys_fw_metadata_t Struct Reference	178
5.89.1	Detailed Description	178
5.89.2	Field Documentation	179
5.89.2.1	active_boot_app	179
5.89.2.2	boot_app_id	179
5.89.2.3	boot_app_ver_status	179
5.89.2.4	boot_app_version	179
5.89.2.5	boot_last_row	179
5.89.2.6	boot_seq	179
5.89.2.7	fw_checksum	179
5.89.2.8	fw_entry	179
5.89.2.9	fw_size	180
5.89.2.10	fw_version	180
5.89.2.11	metadata_valid	180
5.89.2.12	reserved1	180
5.89.2.13	reserved2	180
5.90	pd_do_t::TBT_CBL_VDO Struct Reference	180
5.90.1	Detailed Description	180
5.90.2	Field Documentation	181
5.90.2.1	b22_retimer_cbl	181
5.90.2.2	cbl_gen	181
5.90.2.3	cbl_speed	181
5.90.2.4	cbl_type	181
5.90.2.5	intel_mode	181
5.90.2.6	link_training	181
5.90.2.7	rsvd1	181
5.91	pd_do_t::TBT_UFP_VDO Struct Reference	181
5.91.1	Detailed Description	182
5.91.2	Field Documentation	182
5.91.2.1	adapter	182
5.91.2.2	intel_mode	182
5.91.2.3	rsvd0	182

5.91.2.4	rsvd1	182
5.91.2.5	vpro_supp	182
5.92	pd_do_t::TBT_VDO Struct Reference	182
5.92.1	Detailed Description	183
5.92.2	Field Documentation	183
5.92.2.1	adapter	183
5.92.2.2	b22_retimer_cbl	183
5.92.2.3	cable_active	183
5.92.2.4	cbl_gen	183
5.92.2.5	cbl_speed	183
5.92.2.6	cbl_type	184
5.92.2.7	intel_mode	184
5.92.2.8	link_training	184
5.92.2.9	rsvd1	184
5.92.2.10	rsvd2	184
5.92.2.11	vpro_dock_host	184
5.93	tbthost_cfg_settings_t Struct Reference	184
5.93.1	Detailed Description	185
5.93.2	Field Documentation	185
5.93.2.1	host_support	185
5.93.2.2	hpd_handling	185
5.93.2.3	non_tbt_mux	185
5.93.2.4	pref_data_role	185
5.93.2.5	pref_pwr_role	185
5.93.2.6	rsvd0	185
5.93.2.7	sbu_config	185
5.93.2.8	table_len	186
5.93.2.9	tbt_ctrlr_type	186
5.93.2.10	usb3_support	186
5.93.2.11	usb4_support	186
5.93.2.12	vpro_capable	186
5.94	pd_do_t::UFP_VDO_1 Struct Reference	186
5.94.1	Detailed Description	186
5.94.2	Field Documentation	186
5.94.2.1	alt_modes	187
5.94.2.2	dev_cap	187
5.94.2.3	rsvd0	187
5.94.2.4	rsvd1	187
5.94.2.5	usb_sig	187
5.94.2.6	vdo_version	187

5.95	pd_do_t::USTD_QC_4_0_HDR Struct Reference	187
5.95.1	Detailed Description	187
5.95.2	Field Documentation	188
5.95.2.1	cmd_0	188
5.95.2.2	cmd_1	188
5.95.2.3	svid	188
5.95.2.4	vdm_type	188
5.96	pd_do_t::USTD_VDM_HDR Struct Reference	188
5.96.1	Detailed Description	188
5.96.2	Field Documentation	188
5.96.2.1	cmd	189
5.96.2.2	cmd_type	189
5.96.2.3	rsvd1	189
5.96.2.4	seq_num	189
5.96.2.5	svid	189
5.96.2.6	vdm_type	189
5.96.2.7	vdm_ver	189
5.97	uvp_settings_t Struct Reference	189
5.97.1	Detailed Description	190
5.97.2	Field Documentation	190
5.97.2.1	debounce	190
5.97.2.2	retry_cnt	190
5.97.2.3	table_len	190
5.97.2.4	threshold	190
5.98	pd_do_t::VAR_SNK Struct Reference	190
5.98.1	Detailed Description	190
5.98.2	Field Documentation	191
5.98.2.1	max_voltage	191
5.98.2.2	min_voltage	191
5.98.2.3	op_current	191
5.98.2.4	supply_type	191
5.99	pd_do_t::VAR_SRC Struct Reference	191
5.99.1	Detailed Description	191
5.99.2	Field Documentation	191
5.99.2.1	max_current	192
5.99.2.2	max_voltage	192
5.99.2.3	min_voltage	192
5.99.2.4	supply_type	192
5.100	vconn_ocp_settings_t Struct Reference	192
5.100.1	Detailed Description	192

5.100.2	Field Documentation	192
5.100.2.1	debounce	192
5.100.2.2	retry_cnt	193
5.100.2.3	table_len	193
5.100.2.4	threshold	193
5.101	vdm_msg_info_t Struct Reference	193
5.101.1	Detailed Description	193
5.101.2	Field Documentation	193
5.101.2.1	sop_type	193
5.101.2.2	vdm_header	193
5.101.2.3	vdo	194
5.101.2.4	vdo_numb	194
5.102	vdm_resp_t Struct Reference	194
5.102.1	Detailed Description	194
5.102.2	Field Documentation	194
5.102.2.1	do_count	194
5.102.2.2	no_resp	194
5.102.2.3	resp_buf	194
6	File Documentation	195
6.1	app/alt_mode/alt_mode_hw.h File Reference	195
6.1.1	Detailed Description	196
6.1.2	Macro Definition Documentation	196
6.1.2.1	HPD_DISABLE_CMD	196
6.1.2.2	HPD_ENABLE_CMD	196
6.1.2.3	NO_DATA	196
6.1.3	Enumeration Type Documentation	196
6.1.3.1	alt_mode_hw_t	196
6.1.3.2	mux_poll_status_t	197
6.1.3.3	mux_select_t	197
6.1.4	Function Documentation	197
6.1.4.1	alt_mode_hw_deinit()	197
6.1.4.2	alt_mode_hw_is_idle()	198
6.1.4.3	alt_mode_hw_set_cbk()	198
6.1.4.4	alt_mode_hw_sleep()	198
6.1.4.5	alt_mode_hw_wakeup()	199
6.1.4.6	dp_snk_get_hpd_state()	199
6.1.4.7	eval_app_alt_hw_cmd()	199
6.1.4.8	eval_hpd_cmd()	200
6.1.4.9	eval_mux_cmd()	200

6.1.4.10	get_mux_state()	200
6.1.4.11	ignore_mux_changes()	201
6.1.4.12	set_mux()	201
6.2	app/alt_mode/alt_modes_mngr.h File Reference	201
6.2.1	Detailed Description	203
6.2.2	Macro Definition Documentation	203
6.2.2.1	ALT_MODE_EVT_DATA_IDX	203
6.2.2.2	ALT_MODE_EVT_IDX	204
6.2.2.3	ALT_MODE_EVT_SIZE	204
6.2.2.4	AM_SVID_CONFIG_OFFSET_IDX	204
6.2.2.5	AM_SVID_CONFIG_SIZE_IDX	204
6.2.2.6	ATCH_TGT	204
6.2.2.7	CABLE	204
6.2.2.8	CBL_DIR_SUPP_MASK	204
6.2.2.9	DFP_ALT_MODE_HPI_OFFSET	204
6.2.2.10	EMPTY_VDO	204
6.2.2.11	EN_FLAG_MASK	205
6.2.2.12	EXIT_ALL_MODES	205
6.2.2.13	FULL_MASK	205
6.2.2.14	IS_FLAG_CHECKED	205
6.2.2.15	MAX_RETRY_CNT	205
6.2.2.16	MAX_SUPP_ALT_MODES	205
6.2.2.17	MODE_NOT_SUPPORTED	205
6.2.2.18	NO_DATA	205
6.2.2.19	NONE_MODE_MASK	205
6.2.2.20	NONE_VDO	206
6.2.2.21	REMOVE_FLAG	206
6.2.2.22	SET_FLAG	206
6.2.2.23	UFP_ALT_MODE_HPI_MASK	206
6.2.2.24	VDM_HDR	206
6.2.2.25	VDO_START_IDX	206
6.2.3	Typedef Documentation	206
6.2.3.1	alt_mode_app_cbk_t	206
6.2.3.2	alt_mode_cbk_t	207
6.2.4	Enumeration Type Documentation	207
6.2.4.1	alt_mode_app_cmd_t	207
6.2.4.2	alt_mode_app_evt_t	207
6.2.4.3	alt_mode_mngr_state_t	208
6.2.4.4	alt_mode_state_t	208
6.2.4.5	fail_status_t	208

6.2.5	Function Documentation	209
6.2.5.1	alt_mode_get_status()	209
6.2.5.2	alt_mode_layer_reset()	209
6.2.5.3	alt_mode_mngr_exit_all()	209
6.2.5.4	alt_mode_mngr_reset_info()	210
6.2.5.5	alt_mode_mngr_sleep()	210
6.2.5.6	alt_mode_mngr_wakeup()	210
6.2.5.7	eval_app_alt_mode_cmd()	211
6.2.5.8	eval_rec_vdm()	211
6.2.5.9	form_alt_mode_event()	211
6.2.5.10	get_alt_mode_numb()	212
6.2.5.11	get_alt_modes_config_svid_idx()	212
6.2.5.12	get_custom_svid()	213
6.2.5.13	get_mode_info()	213
6.2.5.14	get_svid_from_idx()	213
6.2.5.15	get_vdm_buff()	214
6.2.5.16	is_alt_mode_mngr_idle()	214
6.2.5.17	is_svid_supported()	214
6.2.5.18	reg_alt_mode_mngr()	215
6.2.5.19	reset_alt_mode_info()	215
6.2.5.20	set_alt_mode_mask()	215
6.2.5.21	set_custom_svid()	216
6.2.5.22	vdm_task_mngr_alt_mode_process()	216
6.3	app/alt_mode/custom_hpi_vid.h File Reference	216
6.3.1	Detailed Description	217
6.3.2	Macro Definition Documentation	217
6.3.2.1	HPI_AM_SVID	217
6.3.2.2	MAX_HPI_AM_VDO_NUMB	217
6.3.3	Enumeration Type Documentation	217
6.3.3.1	hpi_am_state_t	217
6.3.4	Function Documentation	217
6.3.4.1	reg_hpi_modes()	218
6.4	app/alt_mode/dp_sid.h File Reference	218
6.4.1	Detailed Description	219
6.4.2	Macro Definition Documentation	219
6.4.2.1	DFP_D_CONN	219
6.4.2.2	DP_1_3_SIGNALING	219
6.4.2.3	DP_ALLOWED_MUX_CONFIG_EVT	220
6.4.2.4	DP_ALT_MODE_ID	220
6.4.2.5	DP_APP_CFG_CMD	220

6.4.2.6	DP_APP_CFG_CMD_MAX_NUMB	220
6.4.2.7	DP_APP_CFG_USB_IDX	220
6.4.2.8	DP_APP_VCONN_SWAP_CFG_CMD	220
6.4.2.9	DP_CFG_CMD_ACK_MASK	220
6.4.2.10	DP_CONFIG_SELECT	220
6.4.2.11	DP_CONFIG_SELECT_DPSRC	220
6.4.2.12	DP_DFP_D_CONFIG_C	221
6.4.2.13	DP_DFP_D_CONFIG_D	221
6.4.2.14	DP_DFP_D_CONFIG_E	221
6.4.2.15	DP_DFP_D_CONFIG_F	221
6.4.2.16	DP_HPD_STATE_MASK	221
6.4.2.17	DP_INVALID_CFG	221
6.4.2.18	DP_MAX_IRQ_SIZE	221
6.4.2.19	DP_MUX_CTRL_CMD	221
6.4.2.20	DP_QUEUE_EMPTY_INDEX	221
6.4.2.21	DP_QUEUE_FULL_INDEX	222
6.4.2.22	DP_QUEUE_STATE_SIZE	222
6.4.2.23	DP_SINK_CTRL_CMD	222
6.4.2.24	DP_STATUS_UPDATE_EVT	222
6.4.2.25	DP_SVID	222
6.4.2.26	DP_UFP_MAX_QUEUE_SIZE	222
6.4.2.27	DP_USB_SS_CONFIG	222
6.4.2.28	DP_VDO_IDX	222
6.4.2.29	GET_HPD_IRQ_STAT	222
6.4.2.30	HPD_HIGH_IRQ_HIGH	223
6.4.2.31	HPD_HIGH_IRQ_LOW	223
6.4.2.32	HPD_IRQ_BIT_POS	223
6.4.2.33	HPD_LOW_IRQ_HIGH	223
6.4.2.34	HPD_LOW_IRQ_LOW	223
6.4.2.35	HPD_STATE_BIT_POS	223
6.4.2.36	MAX_DP_VDO_NUMB	223
6.4.2.37	STATUS_UPDATE_VDO	223
6.4.2.38	UFP_D_CONN	223
6.4.2.39	USB_CONFIG_SELECT	224
6.4.3	Enumeration Type Documentation	224
6.4.3.1	dp_conn_t	224
6.4.3.2	dp_port_cap_t	224
6.4.3.3	dp_stat_bm_t	224
6.4.3.4	dp_state_t	225
6.4.4	Function Documentation	225

6.4.4.1	reg_dp_modes()	225
6.5	app/alt_mode/vdm_task_mngr.h File Reference	225
6.5.1	Detailed Description	226
6.5.2	Macro Definition Documentation	227
6.5.2.1	DATA_RST_RETRY_NUMB	227
6.5.2.2	MAX_CABLE_SVID_SUPP	227
6.5.2.3	MAX_DISC_SVID_COUNT	227
6.5.2.4	MAX_SVID_VDO_SUPP	227
6.5.2.5	PD_DISC_ID_AMA_VDO_IDX	227
6.5.2.6	PD_SVID_ID_HDR_VDO_START_IDX	227
6.5.2.7	STD_SVID	227
6.5.2.8	USB4_EUDO_HOST_PARAM_SHIFT	227
6.5.3	Enumeration Type Documentation	227
6.5.3.1	usb4_flag_t	228
6.5.3.2	vdm_evt_t	228
6.5.3.3	vdm_task_t	228
6.5.4	Function Documentation	229
6.5.4.1	enable_vdm_task_mngr()	229
6.5.4.2	enter_usb4()	229
6.5.4.3	is_ufp_disc_started()	229
6.5.4.4	is_vdm_task_idle()	230
6.5.4.5	usb4_update_data_status()	230
6.5.4.6	vdm_get_disc_id_resp()	230
6.5.4.7	vdm_get_disc_svid_resp()	231
6.5.4.8	vdm_task_mngr()	231
6.5.4.9	vdm_task_mngr_deinit()	231
6.5.4.10	vdm_task_mngr_exit_modes()	232
6.5.5	Variable Documentation	232
6.5.5.1	modal_op_support	232
6.6	app/app.h File Reference	232
6.6.1	Detailed Description	235
6.6.2	Macro Definition Documentation	235
6.6.2.1	ADC_VBUS_MIN_OVP_LEVEL	235
6.6.2.2	APP_AME_TIMEOUT_TIMER_PERIOD	235
6.6.2.3	APP_AUTO_DR_SWAP_TRY_PERIOD	235
6.6.2.4	APP_BAD_SINK_TIMEOUT_TIMER_PERIOD	236
6.6.2.5	APP_BB_ON_TIMER_PERIOD	236
6.6.2.6	APP_BC_AFC_DETECT_TIMER_PERIOD	236
6.6.2.7	APP_BC_APPLE_DETECT_TIMER_PERIOD	236
6.6.2.8	APP_BC_CDP_SM_TIMER_PERIOD	236

6.6.2.9	APP_BC_DCP_DETECT_TIMER_PERIOD	236
6.6.2.10	APP_BC_DP_DM_DEBOUNCE_TIMER_PERIOD	236
6.6.2.11	APP_BC_GLITCH_BC_DONE_TIMER_PERIOD	236
6.6.2.12	APP_BC_GLITCH_DM_HIGH_TIMER_PERIOD	236
6.6.2.13	APP_BC_SINK_CONTACT_STABLE_TIMER_PERIOD	237
6.6.2.14	APP_BC_V_NEW_REQUEST_TIMER_PERIOD	237
6.6.2.15	APP_BC_VBUS_CYCLE_TIMER_PERIOD	237
6.6.2.16	APP_BC_VDMSRC_EN_DIS_PERIOD	237
6.6.2.17	APP_BC_VDP_DM_SRC_ON_PERIOD	237
6.6.2.18	APP_CABLE_POWER_UP_DELAY	237
6.6.2.19	APP_CABLE_VDM_START_DELAY	237
6.6.2.20	APP_CBL_DISC_TIMER_PERIOD	237
6.6.2.21	APP_DB_SNK_FET_DIS_DELAY_TIMER_PERIOD	237
6.6.2.22	APP_DR_SWAP_PENDING	238
6.6.2.23	APP_FAULT_RECOVERY_MAX_WAIT	238
6.6.2.24	APP_FAULT_RECOVERY_TIMER_PERIOD	238
6.6.2.25	APP_INITIATE_DR_SWAP_TIMER_PERIOD	238
6.6.2.26	APP_INITIATE_PR_SWAP_TIMER_PERIOD	238
6.6.2.27	APP_INITIATE_SEND_IRQ_CLEAR_ACK_PERIOD	238
6.6.2.28	APP_MAX_SWAP_ATTEMPT_COUNT	238
6.6.2.29	APP_OT_DETECTION_TIMER_PERIOD	238
6.6.2.30	APP_OT_HIGH_TEMP	238
6.6.2.31	APP_OT_LOW_TEMP	239
6.6.2.32	APP_OT_ROOM_TEMP	239
6.6.2.33	APP_OT_VBE_25_C_TEMP_ADDR	239
6.6.2.34	APP_OT_VBE_HIGH_TEMP_ADDR	239
6.6.2.35	APP_OT_VBE_LOW_TEMP_ADDR	239
6.6.2.36	APP_PB_VBATT_DEBOUNCE_IN_MS	239
6.6.2.37	APP_PR_SWAP_PENDING	239
6.6.2.38	APP_PSINK_DIS_MONITOR_TIMER_PERIOD	239
6.6.2.39	APP_PSINK_DIS_TIMER_PERIOD	239
6.6.2.40	APP_PSINK_DIS_VBUS_IN_DIS_PERIOD	240
6.6.2.41	APP_PSOURCE_CF_TIMER_PERIOD	240
6.6.2.42	APP_PSOURCE_DIS_EXT_DIS_TIMER_PERIOD	240
6.6.2.43	APP_PSOURCE_SAFE_FET_ON_MONITOR_TIMER_PERIOD	240
6.6.2.44	APP_PSOURCE_VBUS_SET_TIMER_PERIOD	240
6.6.2.45	APP_RESET_VDM_TIMER_PERIOD	240
6.6.2.46	APP_RETIMER_DISABLE_DELAY	240
6.6.2.47	APP_RETIMER_ENABLE_DELAY	240
6.6.2.48	APP_SBU_DELAYED_CONNECT_PERIOD	240

6.6.2.49	APP_UFP_RECOV_VCONN_SWAP_TIMER_PERIOD	241
6.6.2.50	APP_VCONN_RECOVERY_PERIOD	241
6.6.2.51	APP_VCONN_SWAP_PENDING	241
6.6.2.52	APP_VDM_BUSY_TIMER_PERIOD	241
6.6.2.53	APP_VDM_FAIL_RETRY_PERIOD	241
6.6.2.54	CCG_ACTIVITY_TIMER_PERIOD	241
6.6.2.55	ICL_VSYS_STABLE_WAIT_TIME	241
6.6.2.56	OTP_DEBOUNCE_PERIOD	241
6.6.2.57	PB_DEBOUNCE_PERIOD	241
6.6.2.58	RIDGE_INIT_HPD_DEQUEUE_TIMER_PERIOD	242
6.6.2.59	TBT_MODE_EXIT_CHECK_PERIOD	242
6.6.2.60	THROTTLE_DEBOUNCE_PERIOD	242
6.6.2.61	THROTTLE_WAIT_FOR_PD_PERIOD	242
6.6.2.62	TYPE_A_CUR_SENSE_TIMER_PERIOD	242
6.6.2.63	TYPE_A_PWM_STEP_TIMER_PERIOD	242
6.6.2.64	TYPE_A_REG_SWITCH_TIMER_PERIOD	242
6.6.2.65	UCSI_CONNECT_EVENT_PERIOD	242
6.6.3	Typedef Documentation	242
6.6.3.1	mux_poll_fnc_cbk_t	243
6.6.4	Enumeration Type Documentation	243
6.6.4.1	app_nb_sys_pwr_state_t	243
6.6.4.2	app_port_fault_status_mask_t	243
6.6.4.3	app_thermistor_type_t	244
6.6.4.4	sys_hw_error_type_t	244
6.6.5	Function Documentation	244
6.6.5.1	app_bc_12_sm_start()	244
6.6.5.2	app_conf_for_faulty_dev_removal()	244
6.6.5.3	app_connect_change_handler()	245
6.6.5.4	app_contract_handler()	245
6.6.5.5	app_disable_pd_port()	245
6.6.5.6	app_event_handler()	246
6.6.5.7	app_get_callback_ptr()	246
6.6.5.8	app_get_resp_buf()	246
6.6.5.9	app_get_status()	248
6.6.5.10	app_init()	248
6.6.5.11	app_is_host_hpd_virtual()	248
6.6.5.12	app_is_port_enabled()	249
6.6.5.13	app_otp_check_temp()	249
6.6.5.14	app_otp_enable()	249
6.6.5.15	app_otp_status()	250

6.6.5.16	app_ovp_disable()	250
6.6.5.17	app_ovp_enable()	250
6.6.5.18	app_port_fault_count_exceeded()	251
6.6.5.19	app_power_share_init()	251
6.6.5.20	app_sleep()	251
6.6.5.21	app_task()	251
6.6.5.22	app_update_bc_src_support()	252
6.6.5.23	app_update_sys_pwr_state()	252
6.6.5.24	app_uvp_disable()	252
6.6.5.25	app_uvp_enable()	253
6.6.5.26	app_validate_configtable_offsets()	253
6.6.5.27	app_vdm_layer_reset()	253
6.6.5.28	app_wakeup()	254
6.6.5.29	ccg_app_task_init()	254
6.6.5.30	fault_event_handler()	254
6.6.5.31	fault_handler_clear_counts()	254
6.6.5.32	fault_handler_init_vars()	255
6.6.5.33	fault_handler_task()	255
6.6.5.34	mux_ctrl_bb_enable()	255
6.6.5.35	mux_ctrl_init()	256
6.6.5.36	mux_ctrl_set_cfg()	256
6.6.5.37	pd_get_ptr_cfg_sub_tbl()	256
6.6.5.38	sln_pd_event_handler()	257
6.6.5.39	system_sleep()	257
6.6.5.40	system_vconn_ocp_dis()	257
6.6.5.41	system_vconn_ocp_en()	258
6.6.5.42	vbus_discharge_off()	258
6.6.5.43	vbus_discharge_on()	258
6.6.5.44	vbus_get_value()	259
6.6.5.45	vbus_is_present()	259
6.6.5.46	vconn_change_handler()	259
6.6.5.47	vconn_disable()	261
6.6.5.48	vconn_enable()	261
6.6.5.49	vconn_is_present()	261
6.7	app/battery_charging.h File Reference	263
6.7.1	Detailed Description	264
6.7.2	Macro Definition Documentation	264
6.7.2.1	APPLE_AMP_1A	264
6.7.2.2	APPLE_AMP_2_1A	265
6.7.2.3	APPLE_AMP_2_4A	265

6.7.2.4	BC_AMP_LIMIT	265
6.7.2.5	BC_CMP_0_IDX	265
6.7.2.6	BC_CMP_1_IDX	265
6.7.2.7	CCG_POWER_PRECISION_MULT	265
6.7.2.8	QC3_MIN_VOLT	266
6.7.2.9	QC_AMP_12V	266
6.7.2.10	QC_AMP_20V	266
6.7.2.11	QC_AMP_5V	266
6.7.2.12	QC_AMP_9V	266
6.7.2.13	QC_AMP_CONT	266
6.7.2.14	QC_CONT_VOLT_CHANGE_PER_PULSE	266
6.7.3	Enumeration Type Documentation	266
6.7.3.1	bc_apple_brick_id	266
6.7.3.2	bc_apple_id_t	267
6.7.3.3	bc_apple_term	267
6.7.3.4	bc_charge_mode_t	267
6.7.3.5	bc_d_status_t	268
6.7.3.6	bc_port_role_t	268
6.7.3.7	bc_port_type_t	268
6.7.3.8	bc_qc_class_t	269
6.7.3.9	bc_qc_ver_t	269
6.7.3.10	bc_sink_timer_t	269
6.7.3.11	bc_state_t	269
6.7.4	Function Documentation	270
6.7.4.1	bc_afc_form_vi()	270
6.7.4.2	bc_clear_bc_evt()	270
6.7.4.3	bc_fsm()	271
6.7.4.4	bc_get_config()	271
6.7.4.5	bc_get_status()	271
6.7.4.6	bc_init()	272
6.7.4.7	bc_is_active()	272
6.7.4.8	bc_pd_event_handler()	272
6.7.4.9	bc_port_is_cdp()	273
6.7.4.10	bc_set_bc_evt()	273
6.7.4.11	bc_sleep()	273
6.7.4.12	bc_start()	274
6.7.4.13	bc_stop()	274
6.7.4.14	bc_wakeup()	274
6.7.4.15	ccg_get_system_max_pdp()	275
6.7.4.16	qc_set_cf_limit()	275

6.8	app/intel_tbt/bb_retimer.h File Reference	275
6.8.1	Detailed Description	276
6.8.2	Macro Definition Documentation	276
6.8.2.1	BB_CONN_STATE_REG	276
6.8.2.2	BB_DBR_DEBUG_MODE_REG_WRITE_DELAY	277
6.8.2.3	BB_DBR_DEBUG_POLL_DELAY	277
6.8.2.4	BB_DBR_WAKEUP_DELAY	277
6.8.2.5	BB_DEBUG_MODE_CLEAR	277
6.8.2.6	BB_DEBUG_MODE_DEFAULT	277
6.8.2.7	BB_DEBUG_MODE_REG	277
6.8.2.8	BB_DEBUG_MODE_RX_LOCKED	277
6.8.2.9	BB_DEBUG_MODE_SIZE	277
6.8.2.10	BB_DEBUG_MODE_USB_COMPLIANCE	277
6.8.2.11	BB_DEBUGMODE_POLL_COUNT	278
6.8.2.12	BB_IRQ_HPD_MASK	278
6.8.2.13	BB_STATUS_MASK	278
6.8.2.14	BB_STATUS_REG	278
6.8.2.15	BB_STATUS_Sx_ACTIVE	278
6.8.2.16	BB_USB_3_SPEED_MASK	278
6.8.2.17	MAX_NO_OF_RETIMERS	278
6.8.2.18	RETIMER_CFG_AVAILABLE	278
6.8.2.19	RETIMER_CFG_SLAVE_ADDR	278
6.8.2.20	RETIMER_I2C_TIMEOUT	279
6.8.3	Enumeration Type Documentation	279
6.8.3.1	rt_evt_t	279
6.8.4	Function Documentation	279
6.8.4.1	retimer_clr_evt()	279
6.8.4.2	retimer_disable()	279
6.8.4.3	retimer_enable()	280
6.8.4.4	retimer_force_enable()	280
6.8.4.5	retimer_init()	280
6.8.4.6	retimer_is_present()	281
6.8.4.7	retimer_read_wrapper()	281
6.8.4.8	retimer_reg_write()	281
6.8.4.9	retimer_set_compl_mode()	282
6.8.4.10	retimer_set_evt()	282
6.8.4.11	retimer_set_slave_address()	282
6.8.4.12	retimer_sleep_allowed()	283
6.8.4.13	retimer_start_debug_poll()	283
6.8.4.14	retimer_status_update()	283

6.8.4.15	retimer_task()	284
6.8.4.16	retimer_update_is_pending()	284
6.8.4.17	retimer_write_wrapper()	284
6.8.4.18	set_retimer_status()	285
6.9	app/intel_tbt/icl.h File Reference	285
6.9.1	Detailed Description	286
6.9.2	Macro Definition Documentation	286
6.9.2.1	ADP_EDGE_NONE	286
6.9.2.2	ADP_NEGEDGE	286
6.9.2.3	ADP_POSEDGE	286
6.9.2.4	ADP_STATE_INIT	286
6.9.2.5	ICL_ADP_ATTACH_DEBOUNCE_TIME	286
6.9.2.6	ICL_ADP_DETACH_DEBOUNCE_TIME	287
6.9.2.7	ICL_CTRL_CMD_FORCE_TBT_MODE	287
6.9.2.8	ICL_CTRL_CMD_SOC_TO_EVT_DISABLE	287
6.9.2.9	ICL_RFU_EXIT_DURATION	287
6.9.2.10	ICL_STS_REG_FORCE_TBT_MODE	287
6.9.3	Enumeration Type Documentation	287
6.9.3.1	icl_evt_t	287
6.9.3.2	icl_reg_t	287
6.9.4	Function Documentation	288
6.9.4.1	icl_init()	288
6.9.4.2	icl_set_evt()	288
6.9.4.3	icl_sleep_allowed()	288
6.9.4.4	icl_task()	289
6.9.4.5	icl_vsys_is_present()	289
6.10	app/intel_tbt/intel_ridge.h File Reference	289
6.10.1	Detailed Description	290
6.10.2	Function Documentation	290
6.10.2.1	ridge_eval_cmd()	290
6.10.2.2	ridge_force_status_update()	290
6.10.2.3	ridge_set_mux()	290
6.10.2.4	ridge_set_vpro()	291
6.10.2.5	ridge_update_dr()	291
6.10.2.6	tr_hpd_deinit()	292
6.10.2.7	tr_hpd_init()	292
6.10.2.8	tr_hpd_sendevt()	292
6.10.2.9	tr_is_hpd_change()	293
6.11	app/intel_tbt/intel_vid.h File Reference	293
6.11.1	Detailed Description	293

6.11.2	Macro Definition Documentation	294
6.11.2.1	BB_STATUS	294
6.11.2.2	GET_LEGACY_TBT_ADAPTER	294
6.11.2.3	INTEL_VID	294
6.11.2.4	MAX_TBT_VDO_NUMB	294
6.11.2.5	TBT_ALT_MODE_ID	294
6.11.2.6	TBT_EXIT	294
6.11.2.7	TBT_VDO_IDX	294
6.11.2.8	USB2_ENABLE	294
6.11.2.9	VPRO_ALT_MODE_ID	295
6.11.3	Enumeration Type Documentation	295
6.11.3.1	tbt_cbl_gen_t	295
6.11.3.2	tbt_cbl_speed_t	295
6.11.3.3	tbt_state_t	295
6.11.4	Function Documentation	296
6.11.4.1	reg_intel_modes()	296
6.12	app/intel_tbt/ridge_slave.h File Reference	296
6.12.1	Detailed Description	297
6.12.2	Macro Definition Documentation	297
6.12.2.1	ICL_SOC_ACK_TIMEOUT_PERIOD	297
6.12.2.2	PMC_SLAVE_ADDR_PORT0	297
6.12.2.3	PMC_SLAVE_ADDR_PORT1	297
6.12.2.4	RIDGE_CMD_CCG_RESET	297
6.12.2.5	RIDGE_CMD_INT_CLEAR	297
6.12.2.6	RIDGE_IRQ_ACK	298
6.12.2.7	RIDGE_SLAVE_MAX_READ_SIZE	298
6.12.2.8	RIDGE_SLAVE_MAX_WRITE_SIZE	298
6.12.2.9	RIDGE_SLAVE_MIN_WRITE_SIZE	298
6.12.2.10	RIDGE_SLAVE_SCB_CLOCK_FREQ	298
6.12.2.11	RIDGE_SLAVE_SCB_INDEX	298
6.12.3	Enumeration Type Documentation	298
6.12.3.1	ridge_slave_reg_addr_t	298
6.12.4	Function Documentation	299
6.12.4.1	ridge_reg_reset()	299
6.12.4.2	ridge_slave_deinit()	299
6.12.4.3	ridge_slave_init()	299
6.12.4.4	ridge_slave_is_host_connected()	300
6.12.4.5	ridge_slave_set_ocp_status()	300
6.12.4.6	ridge_slave_sleep()	300
6.12.4.7	ridge_slave_soc_timeout_event_control()	300

6.12.4.8	ridge_slave_status_update()	301
6.12.4.9	ridge_slave_task()	301
6.12.4.10	ridge_update_is_pending()	301
6.13	app/pdo.h File Reference	302
6.13.1	Detailed Description	302
6.13.2	Macro Definition Documentation	302
6.13.2.1	APP_PPS_SNK_CONTRACT_PERIOD	302
6.13.2.2	APP_PPS_SNK_CONTRACT_RETRY_PERIOD	302
6.13.3	Function Documentation	302
6.13.3.1	app_update_rdo()	302
6.13.3.2	eval_rdo()	304
6.13.3.3	eval_src_cap()	304
6.14	app/psink.h File Reference	305
6.14.1	Detailed Description	305
6.14.2	Function Documentation	305
6.14.2.1	psnk_disable()	305
6.14.2.2	psnk_enable()	306
6.14.2.3	psnk_set_current()	306
6.14.2.4	psnk_set_voltage()	306
6.15	app/psource.h File Reference	307
6.15.1	Detailed Description	307
6.15.2	Macro Definition Documentation	307
6.15.2.1	PPS_CF_VBUS_DECREMENT_STEP	307
6.15.3	Function Documentation	307
6.15.3.1	psrc_disable()	308
6.15.3.2	psrc_enable()	308
6.15.3.3	psrc_get_voltage()	308
6.15.3.4	psrc_set_current()	309
6.15.3.5	psrc_set_voltage()	309
6.16	app/swap.h File Reference	309
6.16.1	Detailed Description	310
6.16.2	Function Documentation	310
6.16.2.1	eval_dr_swap()	310
6.16.2.2	eval_fr_swap()	310
6.16.2.3	eval_pr_swap()	310
6.16.2.4	eval_vconn_swap()	311
6.17	app/vdm.h File Reference	311
6.17.1	Detailed Description	311
6.17.2	Function Documentation	311
6.17.2.1	eval_enter_usb()	312

6.17.2.2	eval_vdm()	312
6.17.2.3	vdm_data_init()	312
6.17.2.4	vdm_update_data()	313
6.18	hpi/hpi.h File Reference	313
6.18.1	Detailed Description	315
6.18.2	Macro Definition Documentation	315
6.18.2.1	HPI_ADDR_I2C_CFG_FLOAT	315
6.18.2.2	HPI_ADDR_I2C_CFG_HIGH	315
6.18.2.3	HPI_ADDR_I2C_CFG_LOW	315
6.18.3	Typedef Documentation	315
6.18.3.1	hpi_write_cb_t	315
6.18.4	Enumeration Type Documentation	316
6.18.4.1	hpi_boot_prio_conf_t	316
6.18.4.2	hpi_reg_part_t	316
6.18.4.3	hpi_reg_section_t	316
6.18.4.4	hpi_ucsi_control_cmds_t	317
6.18.4.5	hpi_ucsi_status_reg_t	317
6.18.4.6	i2c_owner_t	317
6.18.5	Function Documentation	317
6.18.5.1	hpi_clear_event()	318
6.18.5.2	hpi_deinit()	318
6.18.5.3	hpi_get_port_enable()	318
6.18.5.4	hpi_get_sys_pwr_state()	318
6.18.5.5	hpi_get_ucsi_control()	319
6.18.5.6	hpi_init()	319
6.18.5.7	hpi_init_userdef_regs()	319
6.18.5.8	hpi_is_accessed()	320
6.18.5.9	hpi_is_ec_ready()	320
6.18.5.10	hpi_is_extd_msg_ec_ctrl_enabled()	320
6.18.5.11	hpi_is_vdm_ec_ctrl_enabled()	320
6.18.5.12	hpi_pd_event_handler()	321
6.18.5.13	hpi_reg_enqueue_event()	321
6.18.5.14	hpi_send_fw_ready_event()	322
6.18.5.15	hpi_send_hw_error_event()	322
6.18.5.16	hpi_set_boot_priority_conf()	322
6.18.5.17	hpi_set_ec_interrupt()	323
6.18.5.18	hpi_set_event()	323
6.18.5.19	hpi_set_fixed_slave_address()	323
6.18.5.20	hpi_set_flash_params()	323
6.18.5.21	hpi_set_hpi_version()	324

6.18.5.22 hpi_set_mode_regs()	324
6.18.5.23 hpi_set_no_boot_mode()	325
6.18.5.24 hpi_set_port_event_mask()	325
6.18.5.25 hpi_set_userdef_write_handler()	325
6.18.5.26 hpi_sleep()	326
6.18.5.27 hpi_sleep_allowed()	326
6.18.5.28 hpi_task()	326
6.18.5.29 hpi_update_fw_locations()	326
6.18.5.30 hpi_update_pdo_change()	327
6.18.5.31 hpi_update_versions()	327
6.18.5.32 hpid_get_ec_active_modes()	327
6.18.5.33 ucsi_clear_status_bit()	328
6.18.5.34 ucsi_get_status_bit()	328
6.18.5.35 ucsi_reg_reset()	328
6.18.5.36 ucsi_set_status_bit()	328
6.19 pd_common/dpm.h File Reference	329
6.19.1 Detailed Description	330
6.19.2 Function Documentation	330
6.19.2.1 dpm_clear_fault_active()	330
6.19.2.2 dpm_clear_hard_reset_count()	330
6.19.2.3 dpm_deepsleep()	331
6.19.2.4 dpm_disable()	331
6.19.2.5 dpm_downgrade_pd_port_rev()	331
6.19.2.6 dpm_get_cable_usb_cap()	332
6.19.2.7 dpm_get_def_cable_cap()	332
6.19.2.8 dpm_get_info()	332
6.19.2.9 dpm_get_mux_enable_wait_period()	333
6.19.2.10 dpm_get_ndiscover_identity_count()	333
6.19.2.11 dpm_get_pd_port_status()	333
6.19.2.12 dpm_get_polarity()	333
6.19.2.13 dpm_get_rp_audio_accessory()	334
6.19.2.14 dpm_get_sink_detach_margin()	334
6.19.2.15 dpm_get_sink_detach_voltage()	334
6.19.2.16 dpm_get_snk_wait_cap_period()	335
6.19.2.17 dpm_get_stack_config()	335
6.19.2.18 dpm_get_vbus_voltage()	335
6.19.2.19 dpm_init()	335
6.19.2.20 dpm_is_accessory_mode_disabled()	336
6.19.2.21 dpm_is_idle()	336
6.19.2.22 dpm_is_rdo_valid()	336

6.19.2.23 dpm_is_rp_detach_detect_disabled() 337

6.19.2.24 dpm_pd3_src_rp_flow_control() 337

6.19.2.25 dpm_pd_command() 337

6.19.2.26 dpm_pd_command_ec() 339

6.19.2.27 dpm_pe_stop() 340

6.19.2.28 dpm_pps_task() 340

6.19.2.29 dpm_prot_reset() 340

6.19.2.30 dpm_prot_reset_rx() 341

6.19.2.31 dpm_refresh_snk_cap() 341

6.19.2.32 dpm_refresh_src_cap() 341

6.19.2.33 dpm_send_hard_reset() 342

6.19.2.34 dpm_set_accessory_mode_disabled() 342

6.19.2.35 dpm_set_alert() 342

6.19.2.36 dpm_set_cf() 343

6.19.2.37 dpm_set_chunk_transfer_running() 343

6.19.2.38 dpm_set_delay_src_cap_start() 343

6.19.2.39 dpm_set_fault_active() 344

6.19.2.40 dpm_set_rp_detach_detect_disabled() 344

6.19.2.41 dpm_sleep() 344

6.19.2.42 dpm_start() 344

6.19.2.43 dpm_stop() 346

6.19.2.44 dpm_task() 346

6.19.2.45 dpm_typec_command() 346

6.19.2.46 dpm_typec_deassert_rp_rd() 347

6.19.2.47 dpm_update_def_cable_cap() 347

6.19.2.48 dpm_update_ext_snk_cap() 348

6.19.2.49 dpm_update_ext_src_cap() 348

6.19.2.50 dpm_update_frs_enable() 348

6.19.2.51 dpm_update_mux_enable_wait_period() 349

6.19.2.52 dpm_update_ndiscover_identity_count() 349

6.19.2.53 dpm_update_port_config() 349

6.19.2.54 dpm_update_port_status() 350

6.19.2.55 dpm_update_rp_audio_accessory() 350

6.19.2.56 dpm_update_snk_cap() 351

6.19.2.57 dpm_update_snk_cap_mask() 351

6.19.2.58 dpm_update_snk_max_min() 351

6.19.2.59 dpm_update_snk_wait_cap_period() 352

6.19.2.60 dpm_update_src_cap() 352

6.19.2.61 dpm_update_src_cap_mask() 353

6.19.2.62 dpm_update_swap_response() 353

6.19.2.63 dpm_wakeup()	353
6.19.3 Variable Documentation	353
6.19.3.1 gl_dpm_port_type	353
6.20 pd_common/pd.h File Reference	354
6.20.1 Detailed Description	363
6.20.2 Macro Definition Documentation	363
6.20.2.1 BC_SINK_1_2_MODE_ENABLE_MASK	363
6.20.2.2 BC_SINK_APPLE_MODE_ENABLE_MASK	363
6.20.2.3 BC_SRC_1_2_MODE_ENABLE_MASK	363
6.20.2.4 BC_SRC_AFC_MODE_ENABLE_MASK	363
6.20.2.5 BC_SRC_APPLE_MODE_ENABLE_MASK	363
6.20.2.6 BC_SRC_QC_4_0_MODE_ENABLE_MASK	363
6.20.2.7 BC_SRC_QC_MODE_ENABLE_MASK	364
6.20.2.8 BC_SRC_QC_VER_2_CLASS_A_VAL	364
6.20.2.9 BC_SRC_QC_VER_2_CLASS_B_VAL	364
6.20.2.10 BC_SRC_QC_VER_3_CLASS_A_VAL	364
6.20.2.11 BC_SRC_QC_VER_3_CLASS_B_VAL	364
6.20.2.12 BDO_HDR_IDX	364
6.20.2.13 CC_CHANNEL_1	364
6.20.2.14 CC_CHANNEL_2	364
6.20.2.15 CCG_CC_STAT_DRP_TOGGLE	364
6.20.2.16 CCG_CC_STAT_RD_PRESENT	365
6.20.2.17 CCG_CC_STAT_RP_PRESENT	365
6.20.2.18 CCG_CC_STAT_VCONN_ACTIVE	365
6.20.2.19 CCG_CC_STAT_ZOPEN	365
6.20.2.20 CCG_DPM_ERROR_NO_VCONN	365
6.20.2.21 CCG_DPM_ERROR_NONE	365
6.20.2.22 CCG_FRS_RX_ENABLE_MASK	365
6.20.2.23 CCG_FRS_TX_ENABLE_MASK	365
6.20.2.24 CCG_PD_EXT_PPS_STATUS_SIZE	365
6.20.2.25 CCG_PD_EXT_SNKCAP_BUF_SIZE	366
6.20.2.26 CCG_PD_EXT_SNKCAP_SIZE	366
6.20.2.27 CCG_PD_EXT_SNKCAP_VERS_INDEX	366
6.20.2.28 CCG_PD_EXT_SRCCAP_INP_INDEX	366
6.20.2.29 CCG_PD_EXT_SRCCAP_INP_UNCONSTRAINED	366
6.20.2.30 CCG_PD_EXT_SRCCAP_PDP_INDEX	366
6.20.2.31 CCG_PD_EXT_SRCCAP_SIZE	366
6.20.2.32 CCG_PD_EXT_STATUS_SIZE	366
6.20.2.33 CCG_PD_FIX_SRC_PDO_MASK_REV2	366
6.20.2.34 CCG_PD_FIX_SRC_PDO_MASK_REV3	367

6.20.2.35 CCG_PD_FLAG_CONTRACT_NEG_ACTIVE	367
6.20.2.36 CCG_PD_FLAG_EXPLICIT_CONTRACT	367
6.20.2.37 CCG_PD_FLAG_POWER_SINK	367
6.20.2.38 CCG_PD_FLAG_SRC_READY	367
6.20.2.39 CCG_USB4_EUDO_USB4_EN_MASK	367
6.20.2.40 CCG_USB_MODE_USB2	367
6.20.2.41 CCG_USB_MODE_USB3	367
6.20.2.42 CCG_USB_MODE_USB4	367
6.20.2.43 CERT_STAT_IDX	368
6.20.2.44 CY_VID	368
6.20.2.45 DP_HPD_TYPE_GPIO	368
6.20.2.46 DP_HPD_TYPE_VIRTUAL	368
6.20.2.47 DP_SVID	368
6.20.2.48 DRP_TOGGLE_PERIOD	368
6.20.2.49 GET_DR_SWAP_RESP	368
6.20.2.50 GET_PR_SWAP_RESP	368
6.20.2.51 GET_VCONN_SWAP_RESP	368
6.20.2.52 GIVE_BACK_MASK	369
6.20.2.53 HOST_SBU_CFG_FIXED_POL	369
6.20.2.54 HOST_SBU_CFG_FULL	369
6.20.2.55 HOST_SBU_CFG_PASS_THROUGH	369
6.20.2.56 HPD_RX_ACTIVITY_TIMER_PERIOD_MAX	369
6.20.2.57 HPD_RX_ACTIVITY_TIMER_PERIOD_MIN	369
6.20.2.58 I_OP9A	369
6.20.2.59 I_1A	369
6.20.2.60 I_1P5A	369
6.20.2.61 I_2A	370
6.20.2.62 I_3A	370
6.20.2.63 I_4A	370
6.20.2.64 I_5A	370
6.20.2.65 ID_HEADER_IDX	370
6.20.2.66 ISAFE_0A	370
6.20.2.67 ISAFE_DEF	370
6.20.2.68 MAX_CBL_DSC_ID_COUNT	370
6.20.2.69 MAX_EXTD_MSG_LEGACY_LEN	370
6.20.2.70 MAX_EXTD_PKT_SIZE	371
6.20.2.71 MAX_EXTD_PKT_WORDS	371
6.20.2.72 MAX_HARD_RESET_COUNT	371
6.20.2.73 MAX_MESSAGE_ID	371
6.20.2.74 MAX_NO_OF_DO	371

6.20.2.75 MAX_NO_OF_PDO	371
6.20.2.76 MAX_NO_OF_VDO	371
6.20.2.77 MAX_PR_SWAP_WAIT_COUNT	371
6.20.2.78 MAX_SOP_TYPES	371
6.20.2.79 MAX_SRC_CAP_COUNT	372
6.20.2.80 PD_BIST_CONT_MODE_TIMER_PERIOD	372
6.20.2.81 PD_CBL_DELAY_TIMER_PERIOD	372
6.20.2.82 PD_CBL_DSC_ID_START_TIMER_PERIOD	372
6.20.2.83 PD_CBL_DSC_ID_TIMER_PERIOD	372
6.20.2.84 PD_CBL_POWER_UP_TIMER_PERIOD	372
6.20.2.85 PD_CBL_READY_TIMER_PERIOD	372
6.20.2.86 PD_COLLISION_SRC_COOL_OFF_TIMER_PERIOD	372
6.20.2.87 PD_CUR_PER_UNIT	372
6.20.2.88 PD_CURRENT_PPS_MULTIPLIER	373
6.20.2.89 PD_DATA_RESET_COMPLETION_DELAY	373
6.20.2.90 PD_DATA_RESET_TIMEOUT_PERIOD	373
6.20.2.91 PD_DATA_RESET_TIMER_PERIOD	373
6.20.2.92 PD_DPM_RESP_REC_RESP_PERIOD	373
6.20.2.93 PD_EXTERNALLY_POWERED_BIT_POS	373
6.20.2.94 PD_HARD_RESET_TX_TIMER_PERIOD	373
6.20.2.95 PD_MAX_SRC_CAP_TRY	373
6.20.2.96 PD_NO_RESPONSE_TIMER_PERIOD	373
6.20.2.97 PD_PHY_BUSY_TIMER_PERIOD	374
6.20.2.98 PD_PPS_SRC_TIMER_PERIOD	374
6.20.2.99 PD_PS_HARD_RESET_TIMER_PERIOD	374
6.20.2.100 PD_PS_SNK_TRANSITION_TIMER_PERIOD	374
6.20.2.101 PD_PS_SRC_OFF_TIMER_PERIOD	374
6.20.2.102 PD_PS_SRC_ON_TIMER_PERIOD	374
6.20.2.103 PD_PS_SRC_TRANS_TIMER_PERIOD	374
6.20.2.104 PD_RECEIVER_RESPONSE_TIMER_PERIOD	374
6.20.2.105 PD_SENDER_RESPONSE_TIMER_PERIOD	374
6.20.2.106 PD_SINK_TX_TIMER_PERIOD	375
6.20.2.107 PD_SINK_VBUS_TURN_OFF_TIMER_PERIOD	375
6.20.2.108 PD_SINK_VBUS_TURN_ON_TIMER_PERIOD	375
6.20.2.109 PD_SINK_WAIT_CAP_TIMER_PERIOD	375
6.20.2.110 PD_SOURCE_TRANSITION_TIMER_PERIOD	375
6.20.2.111 PD_SRC_CAP_TIMER_PERIOD	375
6.20.2.112 PD_SRC_RECOVER_TIMER_PERIOD	375
6.20.2.113 PD_SWAP_SRC_START_TIMER_PERIOD	375
6.20.2.114 PD_UFP_VCONN_DISCHARGE_DURATION	375

6.20.2.115	PD_VBUS_TURN_OFF_TIMER_PERIOD	376
6.20.2.116	PD_VBUS_TURN_ON_TIMER_PERIOD	376
6.20.2.117	PD_VCONN_OFF_TIMER_PERIOD	376
6.20.2.118	PD_VCONN_ON_TIMER_PERIOD	376
6.20.2.119	PD_VCONN_REAPPLIED_TIMER_PERIOD	376
6.20.2.120	PD_VCONN_SRC_DISC_TIMER_PERIOD	376
6.20.2.121	PD_VCONN_SWAP_INITIATOR_DELAY_PERIOD	376
6.20.2.122	PD_VCONN_SWAP_INITIATOR_TIMER_PERIOD	376
6.20.2.123	PD_VCONN_TURN_ON_TIMER_PERIOD	376
6.20.2.124	PD_VDM_ENTER_MODE_RESPONSE_TIMER_PERIOD	377
6.20.2.125	PD_VDM_EXIT_MODE_RESPONSE_TIMER_PERIOD	377
6.20.2.126	PD_VDM_RESPONSE_TIMER_PERIOD	377
6.20.2.127	PD_VOLT_PER_UNIT	377
6.20.2.128	PD_VOLT_PER_UNIT_PPS	377
6.20.2.129	PRODUCT_TYPE_VDO_1_IDX	377
6.20.2.130	PRODUCT_TYPE_VDO_2_IDX	377
6.20.2.131	PRODUCT_TYPE_VDO_3_IDX	377
6.20.2.132	PRODUCT_VDO_IDX	377
6.20.2.133	RDO_IDX	378
6.20.2.134	SNK_DETACH_VBUS_POLL_COUNT	378
6.20.2.135	SNK_MIN_MAX_MASK	378
6.20.2.136	SRC_DRP_MIN_DC	378
6.20.2.137	STD_SVID	378
6.20.2.138	STD_VDM_VERSION	378
6.20.2.139	STD_VDM_VERSION_IDX	378
6.20.2.140	STD_VDM_VERSION_REV2	378
6.20.2.141	STD_VDM_VERSION_REV3	378
6.20.2.142	TBT_CTRLR_ALPINE_RIDGE_DUAL	379
6.20.2.143	TBT_CTRLR_ALPINE_RIDGE_SGL	379
6.20.2.144	TBT_CTRLR_TITAN_RIDGE_DUAL	379
6.20.2.145	TBT_CTRLR_TITAN_RIDGE_SGL	379
6.20.2.146	TBT_GEN_3	379
6.20.2.147	TBT_SVID	379
6.20.2.148	TYPEC_ATTACH_WAIT_ENTRY_DELAY_PERIOD	379
6.20.2.149	TYPEC_CC_DEBOUNCE_TIMER_PERIOD	379
6.20.2.150	TYPEC_DRP_TRY_TIMER_PERIOD	379
6.20.2.151	TYPEC_ERROR_RECOVERY_TIMER_PERIOD	380
6.20.2.152	TYPEC_FSM_GENERIC	380
6.20.2.153	TYPEC_FSM_NONE	380
6.20.2.154	TYPEC_PD3_RPCHANGE_DEBOUNCE_PERIOD	380

6.20.2.155	TYPEPEC_PD_DEBOUNCE_TIMER_PERIOD	380
6.20.2.156	TYPEPEC_RD_DEBOUNCE_TIMER_PERIOD	380
6.20.2.157	TYPEPEC_SRC_DETACH_DEBOUNCE_PERIOD	380
6.20.2.158	TYPEPEC_TRY_TIMEOUT_PERIOD	380
6.20.2.159	JFP_NON_PH_ALT_MODE_SUPP_MASK	380
6.20.2.160	JFP_VDO_1_RECFCG_ALT_MODE_PARAM_MASK	381
6.20.2.161	VDM_HEADER_IDX	381
6.20.2.162	SAFE_0V	381
6.20.2.163	SAFE_0V_HARD_RESET	381
6.20.2.164	SAFE_0V_PR_SWAP_SNK_SRC	381
6.20.2.165	SAFE_12V	381
6.20.2.166	SAFE_13V	381
6.20.2.167	SAFE_15V	381
6.20.2.168	SAFE_19V	381
6.20.2.169	SAFE_20V	382
6.20.2.170	SAFE_3_6V	382
6.20.2.171	SAFE_5V	382
6.20.2.172	SAFE_9V	382
6.20.3	Typedef Documentation	382
6.20.3.1	app_resp_cbk_t	382
6.20.3.2	dpm_pd_cmd_cbk_t	382
6.20.3.3	dpm_typepec_cmd_cbk_t	383
6.20.3.4	pd_cbk_t	383
6.20.3.5	pwr_ready_cbk_t	383
6.20.3.6	sink_discharge_off_cbk_t	383
6.20.3.7	vdm_resp_cbk_t	384
6.20.4	Enumeration Type Documentation	384
6.20.4.1	apdo_t	384
6.20.4.2	app_evt_t	384
6.20.4.3	app_fault_mask_t	386
6.20.4.4	app_req_status_t	386
6.20.4.5	app_swap_resp_t	387
6.20.4.6	bist_mode_t	387
6.20.4.7	cbl_term_t	387
6.20.4.8	cbl_type_t	388
6.20.4.9	cbl_vbus_cur_t	388
6.20.4.10	ctrl_msg_t	388
6.20.4.11	data_msg_t	389
6.20.4.12	data_reset_state_t	389
6.20.4.13	dpm_pd_cmd_t	390

6.20.4.14 dpm_typec_cmd_resp_t	390
6.20.4.15 dpm_typec_cmd_t	391
6.20.4.16 extd_msg_t	391
6.20.4.17 fr_swap_supp_t	392
6.20.4.18 intel_pf_type_t	392
6.20.4.19 pd_ams_type	392
6.20.4.20 pd_cable_reset_reason_t	392
6.20.4.21 pd_contract_status_t	393
6.20.4.22 pd_devtype_t	393
6.20.4.23 pd_emca_sr_reason_t	393
6.20.4.24 pd_err_recov_reason_t	394
6.20.4.25 pd_hard_reset_reason_t	394
6.20.4.26 pd_msg_class_t	394
6.20.4.27 pd_rev_t	395
6.20.4.28 pd_soft_reset_reason_t	395
6.20.4.29 pdo_sel_alg_t	395
6.20.4.30 pdo_t	396
6.20.4.31 pe_cbl_state_t	396
6.20.4.32 pe_fsm_state_t	396
6.20.4.33 peak_cur_cap_t	397
6.20.4.34 port_role_t	398
6.20.4.35 port_type_t	398
6.20.4.36 rd_cc_status_t	398
6.20.4.37 rdo_type_t	399
6.20.4.38 resp_status_t	399
6.20.4.39 rp_cc_status_t	399
6.20.4.40 rp_term_t	399
6.20.4.41 sop_t	400
6.20.4.42 std_vdm_cmd_t	400
6.20.4.43 std_vdm_cmd_type_t	400
6.20.4.44 std_vdm_prod_t	401
6.20.4.45 std_vdm_ver_t	401
6.20.4.46 try_src_snk_t	401
6.20.4.47 typec_fsm_state_t	402
6.20.4.48 usb_data_sig_t	402
6.20.4.49 usb_dev_cap_t	403
6.20.4.50 usb_host_cap_t	403
6.20.4.51 usb_role_t	403
6.20.4.52 usb_sig_supp_t	403
6.20.4.53 vdm_ams_t	404

6.20.4.54 vdm_type_t	404
6.20.5 Function Documentation	404
6.20.5.1 get_pd_config()	404
6.20.5.2 get_pd_port_config()	405
6.20.5.3 pd_get_ptr_auto_cfg_tbl()	405
6.20.5.4 pd_get_ptr_bat_chg_tbl()	405
6.20.5.5 pd_get_ptr_bb_tbl()	406
6.20.5.6 pd_get_ptr_chg_cfg_tbl()	406
6.20.5.7 pd_get_ptr_host_cfg_tbl()	406
6.20.5.8 pd_get_ptr_icl_tgl_cfg_tbl()	407
6.20.5.9 pd_get_ptr_ocp_tbl()	407
6.20.5.10 pd_get_ptr_otp_tbl()	407
6.20.5.11 pd_get_ptr_ovp_tbl()	408
6.20.5.12 pd_get_ptr_pwr_tbl()	408
6.20.5.13 pd_get_ptr_rcp_tbl()	409
6.20.5.14 pd_get_ptr_scp_tbl()	409
6.20.5.15 pd_get_ptr_tbthost_cfg_tbl()	409
6.20.5.16 pd_get_ptr_type_a_chg_cfg_tbl()	410
6.20.5.17 pd_get_ptr_type_a_pwr_tbl()	410
6.20.5.18 pd_get_ptr_uvp_tbl()	410
6.20.5.19 pd_get_ptr_vconn_ocp_tbl()	411
6.20.5.20 pd_is_msg()	411
6.21 pd_common/pd_policy_engine.h File Reference	411
6.21.1 Detailed Description	412
6.21.2 Function Documentation	412
6.21.2.1 get_pe_state_buf()	412
6.21.2.2 get_spec_rev_determined()	412
6.21.2.3 pe_clear_hard_reset_count()	413
6.21.2.4 pe_disabled()	413
6.21.2.5 pe_fsm()	413
6.21.2.6 pe_get_pps_status()	413
6.21.2.7 pe_init()	414
6.21.2.8 pe_is_busy()	414
6.21.2.9 pe_push_to_buf()	414
6.21.2.10 pe_start()	414
6.21.2.11 pe_stop()	415
6.22 pd_common/pd_protocol.h File Reference	415
6.22.1 Detailed Description	416
6.22.2 Function Documentation	416
6.22.2.1 pd_prot_dis_bist_cm2()	416

6.22.2.2	pd_prot_dis_bist_test_data()	416
6.22.2.3	pd_prot_en_bist_cm2()	417
6.22.2.4	pd_prot_en_bist_test_data()	417
6.22.2.5	pd_prot_frs_rx_disable()	417
6.22.2.6	pd_prot_frs_rx_enable()	417
6.22.2.7	pd_prot_frs_tx_disable()	418
6.22.2.8	pd_prot_frs_tx_enable()	418
6.22.2.9	pd_prot_get_rx_packet()	418
6.22.2.10	pd_prot_init()	419
6.22.2.11	pd_prot_is_busy()	419
6.22.2.12	pd_prot_refresh_roles()	419
6.22.2.13	pd_prot_reset()	420
6.22.2.14	pd_prot_reset_all()	420
6.22.2.15	pd_prot_reset_rx()	420
6.22.2.16	pd_prot_rx_dis()	421
6.22.2.17	pd_prot_rx_en()	421
6.22.2.18	pd_prot_send_cable_reset()	421
6.22.2.19	pd_prot_send_ctrl_msg()	422
6.22.2.20	pd_prot_send_data_msg()	422
6.22.2.21	pd_prot_send_extd_msg()	423
6.22.2.22	pd_prot_send_hard_reset()	423
6.22.2.23	pd_prot_set_avoid_retry()	423
6.22.2.24	pd_prot_start()	424
6.22.2.25	pd_prot_stop()	424
6.23	pd_common/typec_manager.h File Reference	424
6.23.1	Detailed Description	425
6.23.2	Function Documentation	425
6.23.2.1	typec_assert_rd()	425
6.23.2.2	typec_assert_rp()	425
6.23.2.3	typec_change_rp()	426
6.23.2.4	typec_deepsleep()	426
6.23.2.5	typec_fsm()	426
6.23.2.6	typec_init()	426
6.23.2.7	typec_is_busy()	427
6.23.2.8	typec_start()	427
6.23.2.9	typec_stop()	427
6.24	pd_hal/hal_ccgx.h File Reference	427
6.24.1	Detailed Description	428
6.24.2	Function Documentation	428
6.24.2.1	system_connect_ovp_trip()	428

6.24.2.2	system_disconnect_ovp_trip()	428
6.24.2.3	system_init()	429
6.24.2.4	system_vbus_ocp_dis()	429
6.24.2.5	system_vbus_ocp_en()	429
6.24.2.6	system_vbus_rcp_dis()	430
6.24.2.7	system_vbus_rcp_en()	430
6.24.2.8	system_vbus_scp_dis()	430
6.24.2.9	system_vbus_scp_en()	431
6.24.2.10	vbus_ocp_handler()	431
6.24.2.11	vbus_rcp_handler()	431
6.24.2.12	vbus_scp_handler()	432
6.25	pd_hal/hpd.h File Reference	432
6.25.1	Detailed Description	433
6.25.2	Macro Definition Documentation	433
6.25.2.1	HPD_EVENT_0_POS	433
6.25.2.2	HPD_EVENT_1_POS	433
6.25.2.3	HPD_EVENT_2_POS	433
6.25.2.4	HPD_EVENT_3_POS	433
6.25.2.5	HPD_EVENT_MASK	433
6.25.2.6	HPD_GET_EVENT_0	434
6.25.2.7	HPD_GET_EVENT_1	434
6.25.2.8	HPD_GET_EVENT_2	434
6.25.2.9	HPD_GET_EVENT_3	434
6.25.2.10	HPD_IP_DIRECTION	434
6.25.2.11	HPD_OP_DIRECTION	434
6.25.3	Enumeration Type Documentation	434
6.25.3.1	hpd_event_type_t	434
6.25.4	Function Documentation	435
6.25.4.1	hpd_deinit()	435
6.25.4.2	hpd_receive_get_status()	435
6.25.4.3	hpd_receive_init()	435
6.25.4.4	hpd_rx_sleep_entry()	436
6.25.4.5	hpd_rx_wakeup()	436
6.25.4.6	hpd_sleep_entry()	437
6.25.4.7	hpd_transmit_init()	437
6.25.4.8	hpd_transmit_senddevt()	437
6.25.4.9	hpd_wakeup()	438
6.25.4.10	is_hpd_rx_state_idle()	438
6.26	pd_hal/pdss_hal.h File Reference	438
6.26.1	Detailed Description	444

6.26.2	Macro Definition Documentation	444
6.26.2.1	AUTO_CTRL_MESSAGE_GOODCRC_MASK_CFG	444
6.26.2.2	AUTO_DATA_MESSAGE_GOODCRC_MASK_CFG	444
6.26.2.3	AUTO_EXTD_MESSAGE_GOODCRC_MASK_CFG	444
6.26.2.4	EXPECTED_GOOD_CRC_CLEAR_MASK	444
6.26.2.5	EXPECTED_GOOD_CRC_HDR_DIFF_MASK_REV3	444
6.26.2.6	EXPECTED_GOOD_CRC_HDR_MASK	444
6.26.2.7	PD_MSG_HDR_REV2_IGNORE_MASK	444
6.26.2.8	PDSS_CC_FILT_CYCLES	445
6.26.2.9	PDSS_DRP_HIGH_PERIOD_MS	445
6.26.2.10	PDSS_DRP_HIGH_PERIOD_VAL	445
6.26.2.11	PDSS_DRP_TOGGLE_PERIOD_MS	445
6.26.2.12	PDSS_DRP_TOGGLE_PERIOD_VAL	445
6.26.2.13	PDSS_MAX_RX_MEM_HALF_SIZE	445
6.26.2.14	PDSS_MAX_RX_MEM_SIZE	445
6.26.2.15	PDSS_MAX_TX_MEM_HALF_SIZE	445
6.26.2.16	PDSS_MAX_TX_MEM_SIZE	445
6.26.2.17	PGDO_PD_ISINK_TERMINAL_VAL	446
6.26.2.18	PGDO_PD_ISINK_TERMINAL_VAL_1	446
6.26.2.19	RX_CNT_MAX_VAL	446
6.26.2.20	RX_UI_BOUNDARY_DELTA_VAL	446
6.26.2.21	VBUS_OCP_GPIO_ACTIVE_HIGH	446
6.26.2.22	VBUS_OCP_GPIO_ACTIVE_LOW	446
6.26.2.23	VBUS_OCP_MODE_EXT	446
6.26.2.24	VBUS_OCP_MODE_INT	446
6.26.2.25	VBUS_OCP_MODE_INT_AUTOCTRL	446
6.26.2.26	VBUS_OCP_MODE_INT_SW_DB	447
6.26.2.27	VBUS_OCP_MODE_POLLING	447
6.26.3	Typedef Documentation	447
6.26.3.1	pasc_ptdrv_cont_cbk_t	447
6.26.3.2	pasc_valley_cbk_t	447
6.26.3.3	PD_ADC_CB_T	447
6.26.3.4	pd_cmp_cbk_t	448
6.26.3.5	pd_phy_cbk_t	448
6.26.3.6	pd_supply_change_cbk_t	448
6.26.3.7	vbus_cf_cbk_t	448
6.26.3.8	vbus_load_chg_cbk_t	449
6.26.4	Enumeration Type Documentation	449
6.26.4.1	aux_resistor_config_t	449
6.26.4.2	ccg_refgen_op_t	449

6.26.4.3	ccg_supply_t	450
6.26.4.4	comp_id_t	450
6.26.4.5	comp_tr_id_t	450
6.26.4.6	dpdm_mux_cfg_t	451
6.26.4.7	filter_edge_detect_cfg_t	451
6.26.4.8	filter_id_t	451
6.26.4.9	frs_tx_source_t	452
6.26.4.10	lscsa_app_config_t	452
6.26.4.11	pasc_mode_t	452
6.26.4.12	PD_ADC_ID_T	453
6.26.4.13	PD_ADC_INPUT_T	453
6.26.4.14	PD_ADC_INT_T	453
6.26.4.15	PD_ADC_VREF_T	454
6.26.4.16	pd_fet_dr_t	454
6.26.4.17	pd_phy_evt_t	454
6.26.4.18	sbu_switch_state_t	455
6.26.4.19	vbus_ovp_mode_t	455
6.26.4.20	vbus_uvp_mode_t	455
6.26.5	Function Documentation	455
6.26.5.1	aux_resistor_configure()	456
6.26.5.2	ccg_bc_cdp_en()	456
6.26.5.3	ccg_bc_cdp_sm()	456
6.26.5.4	ccg_bc_dcp_en()	457
6.26.5.5	ccg_bc_dis()	457
6.26.5.6	ccg_bc_is_cdp()	457
6.26.5.7	ccg_config_dp_dm_mux()	457
6.26.5.8	ccg_is_cdp_sm_busy()	458
6.26.5.9	ccg_set_fault_cb()	458
6.26.5.10	get_aux1_resistor_config()	458
6.26.5.11	get_aux2_resistor_config()	460
6.26.5.12	get_sbu1_switch_state()	460
6.26.5.13	get_sbu2_switch_state()	460
6.26.5.14	hpdt_ctrl1_reg_check_start()	461
6.26.5.15	pd_adc_calibrate()	461
6.26.5.16	pd_adc_comparator_ctrl()	461
6.26.5.17	pd_adc_comparator_sample()	462
6.26.5.18	pd_adc_free_run_ctrl()	462
6.26.5.19	pd_adc_get_comparator_status()	463
6.26.5.20	pd_adc_get_vbus_voltage()	463
6.26.5.21	pd_adc_init()	464

6.26.5.22 pd_adc_level_to_volt()	464
6.26.5.23 pd_adc_sample()	465
6.26.5.24 pd_adc_select_vref()	465
6.26.5.25 pd_adc_volt_to_level()	466
6.26.5.26 pd_cf_disable()	466
6.26.5.27 pd_cf_enable()	466
6.26.5.28 pd_cf_get_status()	467
6.26.5.29 pd_cf_mon_disable()	467
6.26.5.30 pd_cf_mon_enable()	467
6.26.5.31 pd_cmp_get_status()	468
6.26.5.32 pd_connect_vbus_div_to_amux()	468
6.26.5.33 pd_disconnect_ra()	468
6.26.5.34 pd_disconnect_vbus_div_from_amux()	469
6.26.5.35 pd_enable_vconn_comp()	469
6.26.5.36 pd_fet_automode_disable()	469
6.26.5.37 pd_fet_automode_enable()	470
6.26.5.38 pd_frs_rx_disable()	470
6.26.5.39 pd_frs_rx_enable()	470
6.26.5.40 pd_frs_tx_disable()	471
6.26.5.41 pd_frs_tx_enable()	471
6.26.5.42 pd_get_vbus_adc_level()	471
6.26.5.43 pd_get_vconn_status()	472
6.26.5.44 pd_hal_abort_auto_toggle()	472
6.26.5.45 pd_hal_cleanup()	472
6.26.5.46 pd_hal_config_auto_toggle()	473
6.26.5.47 pd_hal_dual_fet_config()	473
6.26.5.48 pd_hal_enable_internal_vbus_mon()	474
6.26.5.49 pd_hal_get_vbus_csa_rsense()	474
6.26.5.50 pd_hal_get_vbus_detach_adc()	474
6.26.5.51 pd_hal_get_vbus_detach_input()	474
6.26.5.52 pd_hal_init()	475
6.26.5.53 pd_hal_is_auto_toggle_active()	475
6.26.5.54 pd_hal_is_sink_fet_on()	475
6.26.5.55 pd_hal_measure_ea_volt()	476
6.26.5.56 pd_hal_measure_line_volt()	476
6.26.5.57 pd_hal_measure_vbus()	476
6.26.5.58 pd_hal_measure_vbus_cur()	476
6.26.5.59 pd_hal_measure_vbus_in()	477
6.26.5.60 pd_hal_set_cc_ovp_pending()	477
6.26.5.61 pd_hal_set_fet_drive()	477

6.26.5.62 pd_hal_set_reference()	478
6.26.5.63 pd_hal_set_supply_change_evt_cb()	478
6.26.5.64 pd_hal_set_vbus_csa_rsense()	479
6.26.5.65 pd_hal_set_vbus_detach_params()	479
6.26.5.66 pd_hal_set_vbus_mon_divider()	479
6.26.5.67 pd_hal_typec_sm_restart()	480
6.26.5.68 pd_hal_vconn_ocp_disable()	480
6.26.5.69 pd_hal_vconn_ocp_enable()	480
6.26.5.70 pd_internal_cfet_off()	481
6.26.5.71 pd_internal_cfet_on()	481
6.26.5.72 pd_internal_cfet_soft_start_off()	482
6.26.5.73 pd_internal_cfet_soft_start_on()	482
6.26.5.74 pd_internal_pfet_off()	482
6.26.5.75 pd_internal_pfet_on()	483
6.26.5.76 pd_internal_pfet_soft_start_off()	483
6.26.5.77 pd_internal_pfet_soft_start_on()	484
6.26.5.78 pd_internal_vbus_discharge_off()	484
6.26.5.79 pd_internal_vbus_discharge_on()	484
6.26.5.80 pd_internal_vbus_in_discharge_off()	485
6.26.5.81 pd_internal_vbus_in_discharge_on()	485
6.26.5.82 pd_internal_vbus_load_change_isr_enable()	485
6.26.5.83 pd_internal_vbus_ocp_dis()	486
6.26.5.84 pd_internal_vbus_ocp_en()	486
6.26.5.85 pd_internal_vbus_ovp_dis()	486
6.26.5.86 pd_internal_vbus_ovp_en()	487
6.26.5.87 pd_internal_vbus_rcp_dis()	487
6.26.5.88 pd_internal_vbus_rcp_en()	488
6.26.5.89 pd_internal_vbus_scp_dis()	488
6.26.5.90 pd_internal_vbus_scp_en()	489
6.26.5.91 pd_internal_vbus_uvp_dis()	489
6.26.5.92 pd_internal_vbus_uvp_en()	489
6.26.5.93 pd_is_v5v_supply_on()	490
6.26.5.94 pd_is_vconn_present()	490
6.26.5.95 pd_lscsa_calc_cfg()	491
6.26.5.96 pd_lscsa_cfg()	491
6.26.5.97 pd_pasc_get_valley()	491
6.26.5.98 pd_pasc_lp_disable()	492
6.26.5.99 pd_pasc_lp_ds_allowed()	492
6.26.5.100 pd_pasc_lp_enable()	492
6.26.5.101 pd_pasc_lp_is_active()	493

6.26.5.102pd_pasc_lp_task()	493
6.26.5.103pd_pasc_poll_task()	493
6.26.5.104pd_pasc_ptdrv_cont_det_disable()	494
6.26.5.105pd_pasc_ptdrv_cont_det_enable()	494
6.26.5.106pd_pasc_send_stop_signal()	494
6.26.5.107pd_pasc_set_valley()	495
6.26.5.108pd_pasc_start()	495
6.26.5.109pd_pasc_valley_algo_enable()	495
6.26.5.110pd_pasc_vbus_in_set_volt()	496
6.26.5.111pd_pasc_vbus_in_set_volt_abort()	496
6.26.5.112pd_pasc_vbus_in_set_vsafe_5V()	497
6.26.5.113pd_phy_abort_bist_cm2()	497
6.26.5.114pd_phy_abort_tx_msg()	497
6.26.5.115pd_phy_deepsleep()	498
6.26.5.116pd_phy_dis_unchunked_tx()	498
6.26.5.117pd_phy_en_unchunked_tx()	498
6.26.5.118pd_phy_get_rx_packet()	498
6.26.5.119pd_phy_init()	499
6.26.5.120pd_phy_is_busy()	499
6.26.5.121pd_phy_load_msg()	499
6.26.5.122pd_phy_refresh_roles()	500
6.26.5.123pd_phy_reset_rx_tx_sm()	500
6.26.5.124pd_phy_send_bist_cm2()	501
6.26.5.125pd_phy_send_msg()	501
6.26.5.126pd_phy_send_reset()	501
6.26.5.127pd_phy_wakeup()	502
6.26.5.128pd_remove_internal_fb_res_div()	502
6.26.5.129pd_reset_edge_det()	502
6.26.5.130pd_sample_pfc_comp()	502
6.26.5.131pd_set_pfc_comp()	503
6.26.5.132pd_set_sr_comp()	503
6.26.5.133pd_srgdrv_disable()	504
6.26.5.134pd_srgdrv_enable()	504
6.26.5.135pd_srsense_disable()	504
6.26.5.136pd_srsense_enable()	505
6.26.5.137pd_stop_pfc_comp()	505
6.26.5.138pd_stop_sr_comp()	505
6.26.5.139pd_typec_dis_dpslp_rp()	506
6.26.5.140pd_typec_dis_rd()	506
6.26.5.141pd_typec_dis_rp()	506

6.26.5.142	pd_typec_en_deadbat_rd()	507
6.26.5.143	pd_typec_en_dpslp_rp()	507
6.26.5.144	pd_typec_en_rd()	507
6.26.5.145	pd_typec_en_rp()	508
6.26.5.146	pd_typec_get_cc_status()	508
6.26.5.147	pd_typec_init()	508
6.26.5.148	pd_typec_rd_enable()	509
6.26.5.149	pd_typec_set_polarity()	509
6.26.5.150	pd_typec_snk_update_trim()	509
6.26.5.151	pd_typec_start()	510
6.26.5.152	pd_typec_stop()	510
6.26.5.153	pd_vconn_disable()	510
6.26.5.154	pd_vconn_enable()	511
6.26.5.155	sbu_switch_configure()	511
6.27	scb/i2c.h File Reference	511
6.27.1	Detailed Description	512
6.27.2	Macro Definition Documentation	512
6.27.2.1	I2C_BLOCK_COUNT	513
6.27.2.2	I2C_CLEAR_INTR_MASK	513
6.27.2.3	I2C_CLEAR_INTR_REQUEST_REG	513
6.27.2.4	I2C_SCB_FIFO_SIZE	513
6.27.2.5	I2C_SCB_RX_FIFO_SIZE	513
6.27.2.6	I2C_SCB_TX_FIFO_SIZE	513
6.27.2.7	I2C_SLAVE_ADDR_MASK_DEFAULT	513
6.27.2.8	I2C_SLAVE_TIMER_BASE	513
6.27.2.9	I2C_SLAVE_TIMER_PERIOD	513
6.27.3	Enumeration Type Documentation	514
6.27.3.1	i2c_cb_cmd_t	514
6.27.3.2	i2c_scb_clock_freq_t	514
6.27.3.3	i2c_scb_mode_t	514
6.27.3.4	i2c_scb_state_t	515
6.27.4	Function Documentation	515
6.27.4.1	i2c_reset()	515
6.27.4.2	i2c_scb_deinit()	515
6.27.4.3	i2c_scb_enable_wakeup()	516
6.27.4.4	i2c_scb_init()	516
6.27.4.5	i2c_scb_is_idle()	517
6.27.4.6	i2c_scb_write()	517
6.27.4.7	i2c_slave_ack_ctrl()	518
6.27.4.8	i2c_timer_cb()	518

6.28 system/boot.h File Reference	518
6.28.1 Detailed Description	519
6.28.2 Macro Definition Documentation	519
6.28.2.1 CCG_BL_WAIT_DEFAULT	520
6.28.2.2 CCG_BL_WAIT_MAXIMUM	520
6.28.2.3 CCG_BL_WAIT_MINIMUM	520
6.28.2.4 CCG_BOOT_MODE_RQT_SIG	520
6.28.2.5 CCG_FW1_BOOT_RQT_SIG	520
6.28.2.6 CCG_FW2_BOOT_RQT_SIG	520
6.28.2.7 CCG_FW_METADATA_BOOTSEQ_OFFSET	520
6.28.2.8 CCG_FWMETA_APPID_WAIT_0	520
6.28.2.9 CCG_FWMETA_APPID_WAIT_DEF	520
6.28.2.10 CONFIGTABLE_CHECKSUM_OFFSET	521
6.28.2.11 CONFIGTABLE_CHECKSUM_START	521
6.28.2.12 CONFIGTABLE_SIGNATURE	521
6.28.2.13 CONFIGTABLE_SIZE_OFFSET	521
6.28.2.14 CY_PD_IMG1_FW_STATUS_BIT_MASK	521
6.28.3 Function Documentation	521
6.28.3.1 boot_check_for_valid_fw()	521
6.28.3.2 boot_get_boot_seq()	522
6.28.3.3 boot_get_wait_time()	522
6.28.3.4 boot_handle_validate_fw_cmd()	522
6.28.3.5 boot_jump_to_fw()	522
6.28.3.6 boot_start()	523
6.28.3.7 boot_update_fw_status()	523
6.28.3.8 boot_validate_configtable()	523
6.28.3.9 boot_validate_fw()	524
6.28.3.10 get_boot_mode_reason()	524
6.28.4 Variable Documentation	524
6.28.4.1 gl_img1_fw_metadata	524
6.28.4.2 gl_img1_fw_pseudo_metadata	525
6.28.4.3 gl_img2_fw_metadata	525
6.28.4.4 gl_img2_fw_pseudo_metadata	525
6.28.4.5 gl_img_status	525
6.29 system/ccgx_host_sdk_desc.h File Reference	525
6.29.1 Detailed Description	525
6.30 system/ccgx_version.h File Reference	525
6.30.1 Detailed Description	525
6.30.2 Macro Definition Documentation	525
6.30.2.1 FW_BASE_VERSION	526

6.31	system/flash.h File Reference	526
6.31.1	Detailed Description	527
6.31.2	Macro Definition Documentation	527
6.31.2.1	SROM_API_PARAM_SIZE	527
6.31.3	Typedef Documentation	527
6.31.3.1	flash_cbk_t	527
6.31.4	Enumeration Type Documentation	527
6.31.4.1	flash_app_priority_t	527
6.31.4.2	flash_interface_t	528
6.31.4.3	flash_write_status_t	528
6.31.5	Function Documentation	528
6.31.5.1	flash_access_get_status()	528
6.31.5.2	flash_enter_mode()	529
6.31.5.3	flash_get_access_limits()	529
6.31.5.4	flash_row_clear()	530
6.31.5.5	flash_row_read()	530
6.31.5.6	flash_row_write()	530
6.31.5.7	flash_set_access_limits()	531
6.31.5.8	flash_set_app_priority()	531
6.31.5.9	sflash_row_read()	532
6.31.5.10	sflash_row_write()	532
6.32	system/gpio.h File Reference	532
6.32.1	Detailed Description	534
6.32.2	Macro Definition Documentation	534
6.32.2.1	GPIO_DM_FIELD_MASK	534
6.32.2.2	GPIO_DM_FIELD_SIZE	534
6.32.2.3	GPIO_INT_FIELD_MASK	534
6.32.2.4	GPIO_PORT0_INTR_NO	534
6.32.2.5	GPIO_PORT1_INTR_NO	534
6.32.2.6	GPIO_PORT2_INTR_NO	534
6.32.2.7	GPIO_PORT3_INTR_NO	534
6.32.2.8	GPIO_PORT4_INTR_NO	535
6.32.2.9	GPIO_PORT5_INTR_NO	535
6.32.2.10	GPIO_PORT6_INTR_NO	535
6.32.2.11	HSIOM_FIELD_SHIFT	535
6.32.3	Typedef Documentation	535
6.32.3.1	gpio_intr_cb_t	535
6.32.4	Enumeration Type Documentation	535
6.32.4.1	gpio_dm_t	535
6.32.4.2	gpio_intr_t	537

6.32.4.3	gpio_port_pin_t	537
6.32.4.4	hsiom_mode_t	539
6.32.5	Function Documentation	540
6.32.5.1	gpio_clear_intr()	540
6.32.5.2	gpio_get_intr()	540
6.32.5.3	gpio_hsiom_set_config()	540
6.32.5.4	gpio_int_set_config()	541
6.32.5.5	gpio_read_value()	541
6.32.5.6	gpio_register_intr_cb()	542
6.32.5.7	gpio_set_drv_mode()	542
6.32.5.8	gpio_set_lvttl_mode()	542
6.32.5.9	gpio_set_value()	543
6.32.5.10	hsiom_set_config()	543
6.33	system/instrumentation.h File Reference	544
6.33.1	Detailed Description	544
6.33.2	Enumeration Type Documentation	544
6.33.2.1	inst_evt_t	544
6.33.3	Function Documentation	545
6.33.3.1	instrumentation_init()	545
6.33.3.2	instrumentation_start()	545
6.34	system/srom.h File Reference	545
6.34.1	Detailed Description	545
6.34.2	Macro Definition Documentation	545
6.34.2.1	CALL_MAP	545
6.35	system/status.h File Reference	546
6.35.1	Detailed Description	546
6.35.2	Macro Definition Documentation	546
6.35.2.1	CCG_STATUS_CODE_OFFSET	546
6.35.2.2	CCG_STATUS_TO_HPI_RESPONSE	546
6.35.3	Enumeration Type Documentation	546
6.35.3.1	ccg_status_t	546
6.36	system/system.h File Reference	547
6.36.1	Detailed Description	548
6.36.2	Macro Definition Documentation	548
6.36.2.1	SYS_APP_VERSION_OFFSET	548
6.36.2.2	SYS_BOOT_MODE_RQT_SIG	548
6.36.2.3	SYS_BOOT_TYPE_APP_PRIORITY_POS	548
6.36.2.4	SYS_BOOT_TYPE_FIELD_OFFSET	549
6.36.2.5	SYS_BOOT_TYPE_FW_UPDATE_INTERFACE_MASK	549
6.36.2.6	SYS_BOOT_TYPE_FW_UPDATE_INTERFACE_POS	549

6.36.2.7	SYS_BOOT_TYPE_SECURE_BOOT_MASK	549
6.36.2.8	SYS_BOOT_VERSION_ADDRESS	549
6.36.2.9	SYS_CONFIG_TABLE_SIGN	549
6.36.2.10	SYS_FW_CUSTOM_INFO_OFFSET	549
6.36.2.11	SYS_FW_VERSION_OFFSET	549
6.36.2.12	SYS_INVALID_FW_START_ADDR	549
6.36.2.13	SYS_METADATA_VALID_SIG	550
6.36.2.14	SYS_PSEUDO_METADATA_VALID_SIG	550
6.36.2.15	SYS_SILICON_ID_MASK	550
6.36.2.16	SYS_SILICON_ID_OFFSET	550
6.36.3	Enumeration Type Documentation	550
6.36.3.1	sys_fw_mode_t	550
6.36.4	Function Documentation	550
6.36.4.1	get_silicon_revision()	550
6.36.4.2	sys_get_bcdDevice_version()	551
6.36.4.3	sys_get_boot_version()	551
6.36.4.4	sys_get_custom_info_addr()	551
6.36.4.5	sys_get_device_mode()	552
6.36.4.6	sys_get_fw_img1_start_addr()	552
6.36.4.7	sys_get_fw_img2_start_addr()	552
6.36.4.8	sys_get_img1_fw_version()	552
6.36.4.9	sys_get_img2_fw_version()	553
6.36.4.10	sys_get_recent_fw_image()	553
6.36.4.11	sys_get_silicon_id()	553
6.36.4.12	sys_set_device_mode()	553
6.37	system/timer.h File Reference	554
6.37.1	Detailed Description	555
6.37.2	Macro Definition Documentation	555
6.37.2.1	TIMER_INVALID_ID	555
6.37.2.2	TIMER_INVALID_INDEX	555
6.37.2.3	TIMER_MAX_TIMEOUT	555
6.37.2.4	TIMER_NUM_TIMERS	555
6.37.3	Typedef Documentation	555
6.37.3.1	timer_cb_t	555
6.37.3.2	timer_id_t	555
6.37.4	Function Documentation	556
6.37.4.1	timer_enter_sleep()	556
6.37.4.2	timer_get_count()	556
6.37.4.3	timer_get_multiplier()	556
6.37.4.4	timer_init()	557

6.37.4.5	timer_is_running()	557
6.37.4.6	timer_num_active()	557
6.37.4.7	timer_range_enabled()	558
6.37.4.8	timer_start()	558
6.37.4.9	timer_start_wocb()	558
6.37.4.10	timer_stop()	559
6.37.4.11	timer_stop_all()	559
6.37.4.12	timer_stop_range()	560
6.38	system/timer_id.h File Reference	560
6.38.1	Detailed Description	561
6.38.2	Enumeration Type Documentation	561
6.38.2.1	ccg_timer_id_t	561
6.39	system/utils.h File Reference	566
6.39.1	Detailed Description	567
6.39.2	Macro Definition Documentation	567
6.39.2.1	BUSY_WAIT_US	567
6.39.2.2	BYTE_GET_LOWER_NIBBLE	567
6.39.2.3	BYTE_GET_UPPER_NIBBLE	567
6.39.2.4	DIV_ROUND_NEAREST	567
6.39.2.5	DIV_ROUND_UP	568
6.39.2.6	DWORD_GET_BYTE0	568
6.39.2.7	DWORD_GET_BYTE1	568
6.39.2.8	DWORD_GET_BYTE2	568
6.39.2.9	DWORD_GET_BYTE3	568
6.39.2.10	GET_MAX	568
6.39.2.11	GET_MIN	568
6.39.2.12	MAKE_DWORD	568
6.39.2.13	MAKE_DWORD_FROM_WORD	569
6.39.2.14	MAKE_WORD	569
6.39.2.15	MEM_CMP	569
6.39.2.16	MEM_COPY	569
6.39.2.17	MEM_SET	569
6.39.2.18	REG_FIELD_GET	570
6.39.2.19	REG_FIELD_UPDATE	570
6.39.2.20	REV_BYTE_ORDER	570
6.39.2.21	WORD_GET_LSB	570
6.39.2.22	WORD_GET_MSB	570
6.39.3	Function Documentation	571
6.39.3.1	apply_threshold()	571
6.39.3.2	crc16()	571

6.39.3.3	div_round_up()	571
6.39.3.4	event_group_clear_event()	572
6.39.3.5	event_group_clear_events_by_val()	572
6.39.3.6	event_group_get_event()	572
6.39.3.7	event_group_set_event()	573
6.39.3.8	event_group_set_events_by_val()	573
6.39.3.9	mem_calculate_byte_checksum()	573
6.39.3.10	mem_calculate_dword_checksum()	574
6.39.3.11	mem_calculate_word_checksum()	574
6.39.3.12	mem_copy()	575
6.39.3.13	mem_copy_word()	575
6.39.3.14	mem_set()	575
6.40	ucsi/ucsi.h File Reference	576
6.40.1	Detailed Description	577
6.40.2	Macro Definition Documentation	577
6.40.2.1	ALT_MODES_RECIPIENT_CONNECTOR	577
6.40.2.2	ALT_MODES_RECIPIENT_SOP	577
6.40.2.3	ALT_MODES_RECIPIENT_SOP_DPRIME	577
6.40.2.4	ALT_MODES_RECIPIENT_SOP_PRIME	577
6.40.2.5	CABLE_CURR_3A	578
6.40.2.6	CABLE_CURR_5A	578
6.40.2.7	CABLE_CURR_DFLT	578
6.40.2.8	CABLE_SPEED_10GBPS	578
6.40.2.9	CABLE_SPEED_480MBPS	578
6.40.2.10	CABLE_SPEED_5GBPS	578
6.40.2.11	CABLE_VDO_DIRECTION	578
6.40.2.12	MAX_DPM_CMD_RETRY_COUNT	578
6.40.2.13	POM_BC	578
6.40.2.14	POM_CUR_LEVEL_1_5A	579
6.40.2.15	POM_CUR_LEVEL_3A	579
6.40.2.16	POM_CUR_LEVEL_DEF	579
6.40.2.17	POM_NO_CONSUMER	579
6.40.2.18	POM_PD	579
6.40.2.19	SRC_CUR_LEVEL_1_5A	579
6.40.2.20	SRC_CUR_LEVEL_3A	579
6.40.2.21	SRC_CUR_LEVEL_DEF	579
6.40.2.22	UCSI_BUFFER_SIZE	579
6.40.2.23	UCSI_NOTIFICATION_EN_ALL	580
6.40.2.24	UCSI_NOTIFICATION_EN_REQ	580
6.40.2.25	UCSI_READ_PENDING_EVENT	580

6.40.2.26	UCSI_READ_PENDING_MASK	580
6.40.2.27	UCSI_SET_RP_1_5A	580
6.40.2.28	UCSI_SET_RP_3A	580
6.40.2.29	UCSI_SET_RP_DEF	580
6.40.2.30	UCSI_SET_RP_PPM_DEFAULT	580
6.40.2.31	VDM_DISCOVER_ID	580
6.40.2.32	VDM_DISCOVER_MODES	581
6.40.2.33	VDM_DISCOVER_SVID	581
6.40.3	Function Documentation	581
6.40.3.1	is_port_connect_changed()	581
6.40.3.2	ucsi_configure_send_vdm()	581
6.40.3.3	ucsi_init()	582
6.40.3.4	ucsi_notify()	582
6.40.3.5	ucsi_pd_event_handler()	582
6.40.3.6	ucsi_refresh_conn_cnt()	583
6.40.3.7	ucsi_sleep_allowed()	583
6.40.3.8	ucsi_task()	583
Index		585

Chapter 1

CCGx Firmware Stack: API Reference Guide

1.1 Introduction

USB Type-C is the new USB-IF standard that solves several challenges faced when using today's Type-A and Type-B cables and connectors. USB Type-C uses a slimmer connector (measuring only 2.4-mm in height) to allow for increasing miniaturization of consumer and industrial products. The USB Type-C standard is gaining rapid support by enabling small form-factor, easy-to-use connectors and cables with the ability to transmit multiple protocols and offer power delivery up to 100 W – a significant improvement over the 7.5 W possible using previous standards.

1.1.1 USB Type-C Highlights

- Brand new reversible connector, measuring only 2.4-mm in height.
- Compliant with USB Power Delivery 3.0, providing up to 100 W.
- Double the bandwidth of USB 3.0, increasing to 10 Gbps with SuperSpeedPlus USB3.1.
- Combines multiple protocols in a single cable, including DisplayPort™, PCIe®, Thunderbolt™ and USB4.

1.2 Cypress EZ-PD™ Type-C Controllers

Cypress provides the EZ-PD™ family of USB Type-C controllers which can help implement various Type-C applications. Cypress's Type-C controllers are based on Cypress's PSoC® 4 programmable system-on-chip architecture, which includes programmable analog and digital blocks, an ARM® Cortex®-M0 core and 32 to 128 KB of flash memory.

This product family is driving the industry's first Type-C products with top-tier PC makers, enabling them to bring these USB Type-C benefits to market. Cypress has reference designs readily available for EMCA and dongle applications. These are available online and could be used to speed-up our customers design cycle.

1.2.1 CCGx Product Families

The EZ-PD CCGx product line consists of the following product families:

- EZ-PD™ CCG1: Industry's First Programmable Type-C Port Controller
- EZ-PD™ CCG2: Industry's Smallest Programmable Type-C Port Controller
- EZ-PD™ CCG3: Industry's Most Integrated Type-C Port Controller
- EZ-PD™ CCG4: Industry's First Dual-Port Type-C Port Controller

- EZ-PD™ CCG5: Dual-Port Type-C and PD Port Controller for PCs and Docks
- EZ-PD™ CCG5C: Type-C and PD Port Controller for PCs and Docks
- EZ-PD™ CCG3PA: Power Delivery Solution for Adapters and Chargers
- EZ-PD™ CCG3PA2: Power Delivery Solution for Adapters and Chargers
- EZ-PD™ CCG6: Type-C and PD Port Controller with integrated Load Switch controller

1.3 CCGx Firmware Stack

The CCGx product families are supported by a robust firmware stack which includes:

1. USB Type-C and USB-PD specification compliant PD stack.
2. Drivers for the various hardware blocks on the CCGx controllers.
3. Implementation of a Host Processor Interface (HPI) that allows an external embedded controller to monitor and control the CCGx device operation.
4. Implementation of the DisplayPort Alternate Mode that allows transfer of video signals over the Type-C data lanes.
5. Implementation of the Thunderbolt 3 Alternate Mode that allows the USB-C connection to be used for Thunderbolt, Display and USB functions.
6. Implementation of USB4 capability discovery and negotiation including identification and handling of cable capabilities.
7. I2C based slave interface which reports port connection status to Intel SoCs or Thunderbolt controllers.
8. I2C based master interface to configure retimers as required for various data transfer modes.
9. Implementation of the USB Type-C Connector System Software Interface Specification version 1.1 which allows an OS to manage the Type-C connectors associated with the CCGx controllers.

The CCGx firmware stack is compliant to the:

- USB-PD Specification Revision 3.0, Version 2.0
- Type-C Specification Revision 2.0

This version of the CCGx Firmware Stack supports the CCG6, CCG5C and CCG5 families and is targeted at implementing USB-PD port controllers for desktops and notebooks.

The Host SDK also includes a separate firmware stack for the CCG6DF and CCG6SF families where most of the PD stack functionality has been moved into device ROM.

Please refer to the CCGx Power SDK for stack and code examples for implementing power adapters and chargers using the CCG3PA device family. Please use the CCGx SDK 3.0.2 version for stack and code examples for implementing Type-C video adapters using the CCG3 device family.

1.3.1 Firmware Stack Organization

The CCGx firmware stack is provided in source form with the SDK. The firmware files are organized into the following directory structure:

- **system:** The system folder contains header and source files relating to the CCGx device hardware and registers, boot-loader and flash access functions, low level drivers for the GPIO and USB-PD blocks on the CCGx device, and a soft timer implementation that is used by the firmware stack.

- `pd_hal`: The `pd_hal` folder contains the Hardware Adaptation Layer (HAL) or low level hardware driver for the USB-PD hardware in the CCGx device. The driver functionality include PD block initialization, USB-PD message handling code, interrupt handling and ADC/Comparator configuration.
- `pd_common`: The `pd_common` folder contains the core Type-C and USB-PD stack for the CCGx device. This includes the the Type-C port manager, the USB-PD protocol layer, the USB-PD policy engine and the Device Policy Manager. On devices that dual USB-PD ports, the stack allows both of the ports to function and be managed in a completely independent manner. The PD stack is provided in library form, and hence this folder will only contain the header files that define the stack interfaces.
- `scb`: The `scb` folder contains the driver code for I2C slave mode operation using the Serial Controller Blocks (SCB) on the CCGx device. Since I2C slave mode is the most commonly used interface for CCGx, a specially optimized driver is provided for the same. The SCB driver is provided in library form, and the folder only contains the header files that define the SCB driver interface.
- `hpi`: The `hpi` folder contains the implementation for the Host Processor Interface (HPI) which is an I2C based software protocol that allows an external Embedded Controller (EC) to monitor and control the CCGx device operation. HPI provides a register based interface through which the EC can manage the power contracts, send and receive VDMs and perform flash read/write operations. The HPI implementation is provided in library form and the folder only contains header files that define the interfaces. Please contact Cypress for more details about the HPI interface and its capabilities.
- `app`: The `app` folder contains the top-level application layer functionality that implements the USB-PD controller functions required. This includes functionality such as PDO evaluation and contract negotiation, VDM handling for both DFP and UFP roles, handling of control messages like role swap; and alternate mode discovery and negotiation. The alternate mode specific implementation can be found in the `app/alt_mode` directory.
- `ucsi`: The `ucsi` folder contains the implementation of the USB Type-C Connector System Software Interface which implements registers and commands through which the Operating System can manage the Type-C connectors in the platform.

Chapter 2

CCGx Firmware Architecture

The CCGx firmware solution allows users to implement a variety of USB-PD applications using the CCG devices and a fully tested firmware stack.

The CCGx firmware solution contains the following components:

- **Hardware Abstraction Layer (HAL):** This includes the low level drivers for the various hardware blocks on the CCG device. This includes drivers for the Type-C and USB-PD block, Serial Communication Blocks (SCBs), GPIOs, flash module and timer module.
- **USB Type-C and USB-PD Protocol Stack:** This is the complete USB-PD protocol stack that includes the Type-C and USB-PD port managers, USB-PD protocol layer, the USB-PD policy engine and the device policy manager. The device policy manager is designed to allow all policy decisions to be made at the application level, either on an external Embedded Controller (EC) or in the CCG firmware itself.
- **Firmware update module:** This is a firmware module that allows the device firmware maintained in internal flash to be updated. In Notebook PD port controller applications, the firmware update will be done from the EC side through an I2C interface.
- **Host Processor Interface (HPI):** The Host Processor Interface (HPI) is an I2C based control interface that allows an Embedded Controller (EC) to monitor and control the USB-PD port on the CCG device. The HPI is the means to allow the PC platform to control the PD policy management.
- **Port Management:** This module handles all of the PD port management functions including the algorithm for optimal contract negotiations, source and sink power control, source voltage selection, port role assignment and swap request handling.
- **Alternate Modes:** This module implements the alternate mode handling for CCG as a DFP and UFP. A fully tested implementation of DisplayPort alternate mode with CCG as DFP is provided. The module also allows users to implement their own alternate mode support in both DFP and UFP modes.
- **Low Power:** This module attempts to keep the CCG device in the low power standby mode as often as possible to minimize power consumption.
- **External Hardware Control:** This is an hardware design dependent module which controls the external hardware blocks such as FETs, regulators and Type-C switches.
- **Solution specific tasks:** This is an application layer module where any custom tasks required by the user solution can be implemented.

A block diagram of the CCGx firmware architecture is shown below.

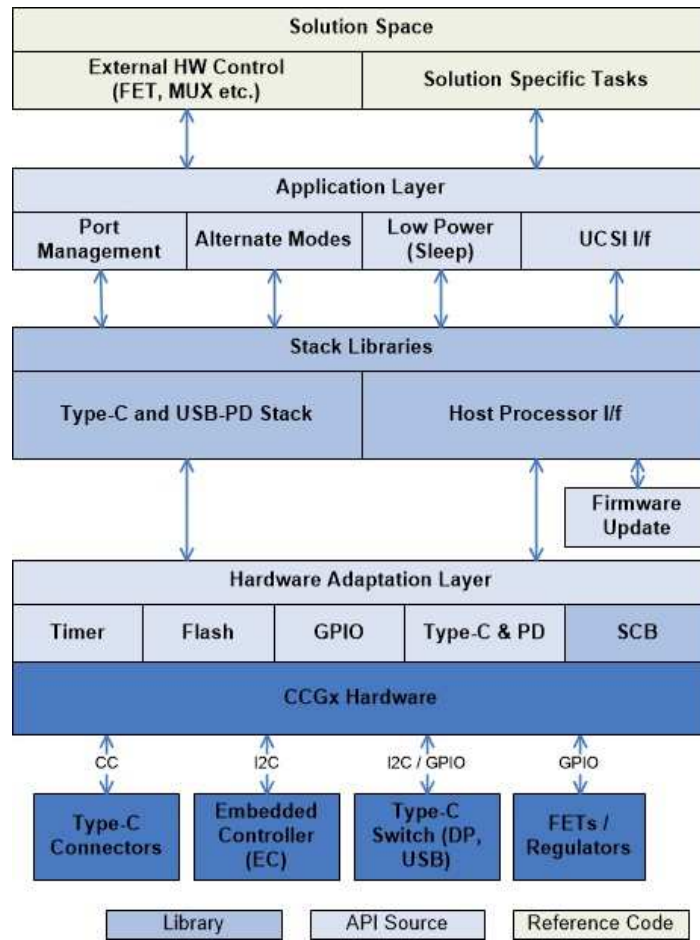


Figure 2.1: CCGx Firmware Stack Block Diagram

2.1 CCGx Firmware Solution Structure

The CCGx firmware solution is broken down into multiple layers:

- **Hardware Adaptation Layer:** The HAL is the lowest driver layer for the CCG device. The HAL code is provided in source form and can be located under the system folder in the CCGx firmware project.
- **USB-PD Stack:** The USB-PD stack includes the Type-C and PD managers, the PD protocol, policy engine and device policy manager blocks. The USB-PD stack is provided in the form of a pre-compiled library.
- **Application Layer:** The application layer includes the PD port management, HPI implementation, alternate modes and low power mode support. The application layer is provided in source form; and can be located under the app and alt_mode folders in the CCGx firmware project.
- **Solution Layer:** The solution layer includes external hardware management, the main task loop and any user application specific functionality. A reference implementation is provided under the solution folder in the CCGx firmware project; and is expected to be customized by users.

The CCGx firmware solution structure is shown below.

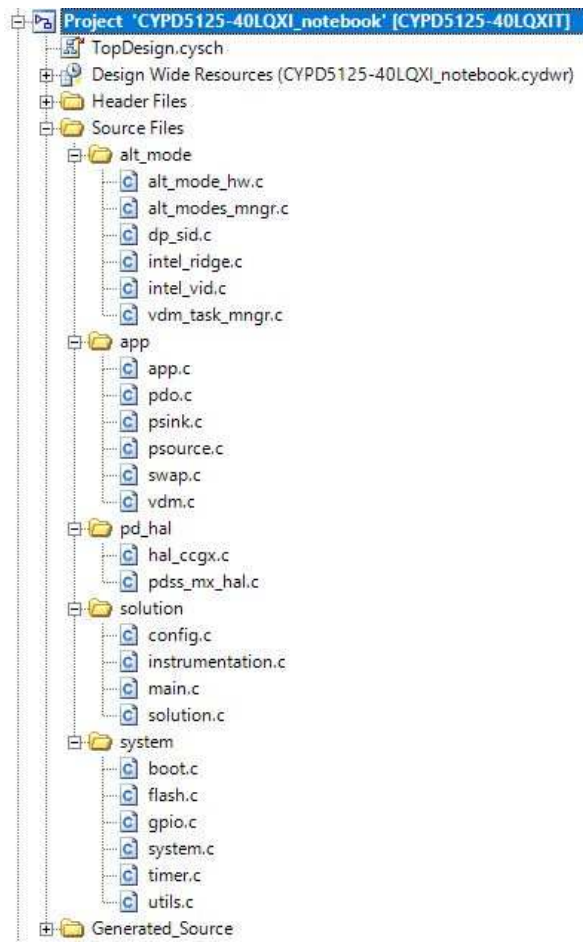


Figure 2.2: CCGx Firmware Solution Structure

2.2 Public API Summary

This document describes all of the data structures and functions that are part of the CCGx firmware stack. Many of these functions are intended to be used internally by the stack layers. The main public interfaces that are expected to be used by user code are summarized in this section.

2.2.1 Device Policy Manager (DPM) API

The Device Policy Manager (DPM) APIs are the public interfaces to the PD stack. These APIs include:

- [dpm_init\(\)](#)
- [dpm_start\(\)](#)
- [dpm_stop\(\)](#)
- [dpm_disable\(\)](#)
- [dpm_deepsleep\(\)](#)
- [dpm_wakeup\(\)](#)
- [dpm_task\(\)](#)
- [dpm_get_info\(\)](#)

- [dpm_update_def_cable_cap\(\)](#)
- [dpm_get_def_cable_cap\(\)](#)
- [dpm_update_snk_wait_cap_period\(\)](#)
- [dpm_get_snk_wait_cap_period\(\)](#)
- [dpm_update_mux_enable_wait_period\(\)](#)
- [dpm_get_mux_enable_wait_period\(\)](#)
- [dpm_pd_command\(\)](#)
- [dpm_typec_command\(\)](#)
- [dpm_update_swap_response\(\)](#)
- [dpm_update_src_cap\(\)](#)
- [dpm_update_src_cap_mask\(\)](#)
- [dpm_update_snk_cap\(\)](#)
- [dpm_update_snk_cap_mask\(\)](#)
- [dpm_update_snk_max_min\(\)](#)
- [dpm_update_port_config\(\)](#)
- [dpm_get_polarity\(\)](#)
- [dpm_typec_deassert_rp_rd\(\)](#)
- [dpm_update_port_status\(\)](#)
- [dpm_get_pd_port_status\(\)](#)
- [dpm_update_frs_enable\(\)](#)
- [dpm_update_ext_src_cap\(\)](#)
- [dpm_prot_reset\(\)](#)
- [dpm_send_hard_reset\(\)](#)
- [dpm_get_sink_detach_margin\(\)](#)
- [dpm_get_sink_detach_voltage\(\)](#)

2.2.2 Host Processor Interface (HPI) API

The Host Processor Interface (HPI) API are functions that initialize the HPI register values and help perform message exchanges between CCG and the Embedded Controller (EC).

- [hpi_init\(\)](#)
- [hpi_task\(\)](#)
- [hpi_reg_enqueue_event\(\)](#)
- [hpi_pd_event_handler\(\)](#)
- [hpi_update_versions\(\)](#)
- [hpi_set_mode_regs\(\)](#)
- [hpi_update_fw_locations\(\)](#)
- [hpi_sleep_allowed\(\)](#)
- [hpi_sleep\(\)](#)
- [hpi_get_port_enable\(\)](#)

2.2.3 Application Layer API

The application layer is where the policy management for the CCGx application are implemented. The default implementations of these functions are based on the configuration table and the parameters provided by EC through the HPI registers.

- [app_init\(\)](#)
- [app_task\(\)](#)
- [app_event_handler\(\)](#)
- [app_get_status\(\)](#)
- [app_sleep\(\)](#)
- [app_wakeup\(\)](#)
- [system_sleep\(\)](#)
- [eval_src_cap\(\)](#)
- [eval_rdo\(\)](#)
- [psnk_set_voltage\(\)](#)
- [psnk_set_current\(\)](#)
- [psnk_enable\(\)](#)
- [psnk_disable\(\)](#)
- [psrc_set_voltage\(\)](#)
- [psrc_set_current\(\)](#)
- [psrc_enable\(\)](#)
- [psrc_disable\(\)](#)
- [vconn_enable\(\)](#)
- [vconn_disable\(\)](#)
- [vconn_is_present\(\)](#)
- [vbus_is_present\(\)](#)
- [vbus_discharge_on\(\)](#)
- [vbus_discharge_off\(\)](#)
- [app_ovp_enable\(\)](#)
- [app_ovp_disable\(\)](#)
- [eval_dr_swap\(\)](#)
- [eval_vconn_swap\(\)](#)
- [eval_pr_swap\(\)](#)
- [vdm_data_init\(\)](#)
- [vdm_update_data\(\)](#)
- [eval_vdm\(\)](#)

2.2.3.1 Solution Layer Functions

The PD stack requires the user to provide a set of functions that provide the stack configuration and also handle operations that are hardware dependent. The following functions are expected to be implemented at the solution layer in user code.

- [mux_ctrl_init\(\)](#)
- [mux_ctrl_set_cfg\(\)](#)
- [sln_pd_event_handler\(\)](#)
- [app_get_callback_ptr\(\)](#)

2.2.4 Alternate Mode Manager API

The following APIs are provided by the CCG Alternate Mode Manager module. Some of these APIs are for use when CCG is a DFP, and the others are for use when CCG is an UFP.

- [enable_vdm_task_mgr\(\)](#)
- [vdm_task_mgr_deinit\(\)](#)
- [vdm_task_mgr\(\)](#)
- [is_vdm_task_idle\(\)](#)
- [eval_rec_vdm\(\)](#)

2.2.5 Hardware Abstraction Layer (HAL) API

This section documents the API provided as part of the Hardware Adaptation Layer (HAL) which provides drivers for various hardware blocks on the CCG device.

2.2.5.1 GPIO API

- [hsiom_set_config\(\)](#)
- [gpio_set_drv_mode\(\)](#)
- [gpio_hsiom_set_config\(\)](#)
- [gpio_int_set_config\(\)](#)
- [gpio_set_value\(\)](#)
- [gpio_read_value\(\)](#)
- [gpio_get_intr\(\)](#)
- [gpio_clear_intr\(\)](#)

2.2.5.2 I2C Driver API

- [i2c_scb_init\(\)](#)
- [i2c_scb_write\(\)](#)
- [i2c_reset\(\)](#)
- [i2c_slave_ack_ctrl\(\)](#)
- [i2c_scb_is_idle\(\)](#)
- [i2c_scb_enable_wakeup\(\)](#)

2.2.5.3 Flash Module API

- [flash_enter_mode\(\)](#)
- [flash_access_get_status\(\)](#)
- [flash_set_access_limits\(\)](#)
- [flash_row_clear\(\)](#)
- [flash_row_write\(\)](#)
- [flash_row_read\(\)](#)

2.2.5.4 Soft Timer API

- [timer_init\(\)](#)
- [timer_start\(\)](#)
- [timer_stop\(\)](#)
- [timer_is_running\(\)](#)
- [timer_stop_all\(\)](#)
- [timer_stop_range\(\)](#)
- [timer_num_active\(\)](#)
- [timer_enter_sleep\(\)](#)

2.2.6 Firmware Update API

These APIs provide the basic boot and firmware upgrade related functions that are used by various firmware update protocol implementations like HPI and CC unstructured command handlers.

- [boot_validate_fw\(\)](#)
- [boot_validate_configtable\(\)](#)
- [get_boot_mode_reason\(\)](#)
- [boot_get_boot_seq\(\)](#)
- [sys_set_device_mode\(\)](#)
- [sys_get_device_mode\(\)](#)

Please see the detailed description of the APIs in the following chapters. You can also refer to the CCGx SDK User Guide document that provides some usage examples and guidelines.

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

- [pd_do_t::ACT_CBL_VDO](#)
Active cable VDO structure as defined by PD 3.0 21
- [pd_do_t::ACT_CBL_VDO_1](#)
Active Cable VDO 1 structure as defined by PD 3.0, Version 1.2 23
- [pd_do_t::ACT_CBL_VDO_2](#)
Active Cable VDO 2 structure as defined by PD 3.0, Version 1.2 25
- [pd_do_t::ADO_ALERT](#)
PD 3.0 Alert Data Object 28
- [alt_cfg_settings_t](#)
Struct to hold the Alt modes settings 29
- [alt_mode_evt_t::ALT_MODE_EVT](#)
Struct containing alternate modes manager event/command 30
- [alt_mode_evt_t::ALT_MODE_EVT_DATA](#)
Struct containing alternate modes manager event/command data 31
- [alt_mode_evt_t](#)
Alt modes manager application event/command structure 32
- [alt_mode_hw_evt_t::ALT_MODE_HW_EVT](#)
Struct containing alternate modes HW event/command 33
- [alt_mode_hw_evt_t](#)
Alt mode HW application event/command structure 33
- [alt_mode_info_t](#)
This structure holds all necessary information for interaction between alt modes manager and selected alternate mode 34
- [alt_mode_reg_info_t](#)
This structure holds all necessary information on Discovery Mode stage for supported alternate mode when alt modes manager registers new alt mode 37
- [app_cbk_t](#)
Struct to hold the application interface. The application is expected to fill the structure with pointers to functions that use the on-board circuitry to accomplish tasks like source/sink power turn on/off. All the functions in this structure must be non-blocking and take minimum execution time 38
- [app_resp_t](#)
Struct to hold response to policy manager. Note: The [app_resp_t](#) structure is only used for responses that have a single DO. This may need to get extended if more DOs are to be supported 42
- [app_status_t](#)
This structure hold all variables related to application layer functionality 42
- [atch_tgt_info_t](#)
This struct holds alternate mode discovery information which is used by alt modes manager . . 49

auto_cfg_settings_t	Struct to hold the automotive charger settings	50
bat_chg_params_t	Struct to hold the battery charging parameters	52
pd_do_t::BAT_SNK	Structure representing a Battery Supply PDO - Sink	53
pd_do_t::BAT_SRC	Structure representing a Battery Supply PDO - Source	54
bb_settings_t	Struct to hold the Billboard settings	55
bc_dp_dm_state_t	Union to hold Dp/Dm status	57
bc_status_t	Struct to define battery charger status	58
pd_do_t::BIST_DO	Structure of a BIST data object	60
cc_state_t	Union to hold CC status	61
ccg_timer_t	Structure encapsulating information relating to a software timer	62
chg_cfg_params_t	Struct to hold the port legacy charging parameters	63
contract_t	Structure to hold PD Contract information	64
custom_alt_cfg_settings_t	Struct to hold the Custom Alt mode settings	65
custom_host_cfg_settings_t	Struct to hold the custom host settings	66
pd_do_t::DFP_VDO	DFP VDO	68
pd_do_t::DP_CONFIG_VDO	DisplayPort configure VDO as defined by VESA spec	69
pd_do_t::DP_STATUS_VDO	DisplayPort status update VDO as defined by VESA spec	70
dpm_pd_cmd_buf_t	Struct to hold PD command buffer	71
dpm_status_t	PD Device Policy Status structure. This structure holds all of the configuration and status information associated with a port on the CCG device. Members of this structure should not be directly modified by any of the application code	73
pd_do_t::ENTERUSB_VDO	Enter USB Data Object	89
pd_extd_hdr_t::EXTD_HDR_T	Extended header broken down into respective fields	91
pd_do_t::FIXED_SNK	Structure representing a Fixed Supply PDO - Sink	92
pd_do_t::FIXED_SRC	Structure representing a Fixed Supply PDO - Source	94
fw_img_status_t	Boot mode reason structure	96
fw_img_status_t::fw_mode_reason_t	97
i2c_scb_config_t	This structure holds all configuration associated with each I2C block	98
icl_tgl_cfg_settings_t	Struct to hold the ICL/TGL settings	99
ocp_settings_t	Struct to hold the OCP settings	101

otp_settings_t	Struct to hold the OTP settings	102
ovp_settings_t	Struct to hold the OVP settings	104
pd_do_t::PAS_CBL_VDO	Passive cable VDO structure as defined by PD 3.0	105
pasc_valley_table_t	The structure contains the valley table to be used to implement the valley algorithm for PAG1S based power adapter solution. The table contains the frequency selection parameters at various voltage and load conditions	107
pd_config_t	Struct to hold the PD device configuration	108
pd_contract_info_t	Structure to hold PD Contract information passed with APP_EVT_PD_CONTRACT_NEGOTIA← TION_COMPLETE event to the application	111
pd_do_t	Union to hold a PD data object. All USB-PD data objects are 4-byte values which are interpreted according to the message type, length and object position. This union represents all possible interpretations of a USB-PD data object	111
pd_extd_hdr_t	Union to hold the PD extended header	118
pd_hdr_t::PD_HDR	PD message header broken down into component fields. Includes 2-byte extended message header	119
pd_hdr_t	Union to hold the PD header defined by the USB-PD specification. Lower 16 bits hold the message header and the upper 16 bits hold the extended message header (where applicable)	121
pd_packet_extd_t	Struct to hold extended PD packets (messages)	121
pd_packet_t	Struct to hold a PD packet	122
pd_port_config_t	PD port-specific configuration data from the configuration table	124
pd_port_status_t::PD_PORT_STAT	Structure containing status bits	133
pd_port_status_t	PD port status as reported to Embedded Controller	136
pd_power_status_t	PD port status corresponding to the Status Data Block (SSDB) See Table 6-39 of USB-PD R3 specification	137
pd_stack_conf_t	Struct used to fetch the PD stack configuration supported by the library that is currently in use	138
pd_status_t	Structure to hold protocol layer status	139
pd_do_t::PPS_SNK	Programmable Power Supply Sink PDO	143
pd_do_t::PPS_SRC	Programmable Power Supply Source PDO	144
prl_cntrs_t	Struct to hold PD protocol message IDs and flags. Separate structures need to be maintained for each packet type (SOP, SOP' and SOP")	146
pwr_params_t	Struct to hold the port power parameters	146
pd_do_t::QC_4_0_DATA_VDO	Structure representing an Unstructured VDM data object as defined by QC 4.0 spec	150
rcp_settings_t	Struct to hold the RCP settings	151

pd_do_t::RDO_BAT	Structure representing a Battery Request Data Object	152
pd_do_t::RDO_BAT_GIVEBACK	Structure representing a Battery Request Data Object with GiveBack	154
pd_do_t::RDO_FIXED_VAR	Structure representing a Fixed or Variable Request Data Object	155
pd_do_t::RDO_FIXED_VAR_GIVEBACK	Structure representing a Fixed or Variable Request Data Object with GiveBack	157
pd_do_t::RDO_GEN	Structure representing a generic Request Data Object	158
pd_do_t::RDO_GEN_GVB	Structure representing a Generic Request Data Object with GiveBack	160
pd_do_t::RDO_PPS	Programmable Request Data Object	162
reg_am_t	Structure to hold the alternate modes SVID and handler	164
scp_settings_t	Struct to hold the SCP settings	164
sensor_data_t	Struct to hold the sensor throttling settings	165
pd_do_t::SRC_GEN	Structure representing a generic source PDO	166
pd_do_t::STD_AMA_VDO	AMA VDO structure as defined by PD 2.0	167
pd_do_t::STD_AMA_VDO_PD3	AMA VDO structure as defined by PD 3.0	169
pd_do_t::STD_CBL_VDO	Cable VDO structure as defined in USB-PD r2.0	170
pd_do_t::STD_CERT_VDO	Cert Stat VDO structure	172
pd_do_t::STD_DP_VDO	DisplayPort Mode VDO as defined by VESA spec	173
pd_do_t::STD_PROD_VDO	Product VDO structure	174
pd_do_t::STD_SVID_RESP_VDO	Discover_SVID response structure	175
pd_do_t::STD_VDM_HDR	Structure representing a Structured VDM Header Data Object	175
pd_do_t::STD_VDM_ID_HDR	Structure representing a Standard ID_HEADER VDO	177
sys_fw_metadata_t	CCGx Firmware metadata structure	178
pd_do_t::TBT_CBL_VDO	Thunderbolt Discover Modes Response Data Object	180
pd_do_t::TBT_UFP_VDO	Thunderbolt UFP Discover Modes Response Data Object	181
pd_do_t::TBT_VDO	Thunderbolt Discover Modes Response Data Object	182
tbthost_cfg_settings_t	Struct to hold Thunderbolt Host related config settings	184
pd_do_t::UFP_VDO_1	UFP VDO #1	186
pd_do_t::USTD_QC_4_0_HDR	Structure representing an Unstructured VDM header data object as defined by QC 4.0 spec	187
pd_do_t::USTD_VDM_HDR	Structure representing an Unstructured VDM header data object as defined by Cypress	188
uvp_settings_t	Struct to hold the UVP settings	189

pd_do_t::VAR_SNK	
Structure representing a Variable Supply PDO - Sink	190
pd_do_t::VAR_SRC	
Structure representing a Variable Supply PDO - Source	191
vconn_ocp_settings_t	
Struct to hold the Vconn OCP settings	192
vdm_msg_info_t	
This struct holds received/sent VDM information which is used by VDM alternative modes managers	193
vdm_resp_t	
Struct to hold response to policy manager	194

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

app/app.h	PD application handler header file	232
app/battery_charging.h	Battery Charging header file	263
app/pdo.h	PDO evaluation and handler definitions	302
app/psink.h	Power Sink (Consumer) manager header file	305
app/psource.h	Power source (Provider) manager header file	307
app/swap.h	Swap request (PR_SWAP, DR_SWAP, VCONN_SWAP) handlers	309
app/vdm.h	Vendor Defined Message (VDM) handler header file	311
app/alt_mode/alt_mode_hw.h	Hardware control for Alternate mode implementation	195
app/alt_mode/alt_modes_mngr.h	Alternate Mode Manager header file	201
app/alt_mode/custom_hpi_vid.h	Custom HPI alternate mode handler header file	216
app/alt_mode/dp_sid.h	DisplayPort alternate mode handler header file	218
app/alt_mode/vdm_task_mngr.h	VDM task manager header file	225
app/intel_tbt/bb_retimer.h	Burnside/Delta bridge retimer interface definitions	275
app/intel_tbt/icl.h	Ice Lake specific logic	285
app/intel_tbt/intel_ridge.h	Intel Alpine/Titan Ridge control interface header file	289
app/intel_tbt/intel_vid.h	Thunderbolt (Intel VID) alternate mode handler header file	293
app/intel_tbt/ridge_slave.h	Alpine/Titan Ridge I2C slave interface header file	296
hpiss/hpi.h	Host Processor Interface (HPI) header file	313
pd_common/dpm.h	Device Policy Manager (DPM) header file	329

pd_common/pd.h	Common definitions and structures used in the CCG USB-PD stack	354
pd_common/pd_policy_engine.h	USB-PD policy engine header file	411
pd_common/pd_protocol.h	USB-PD protocol layer header file	415
pd_common/typec_manager.h	Type-C manager header file	424
pd_hal/hal_ccgx.h	PD and Type-C HAL layer for CCGx device family	427
pd_hal/hpd.h	Hotplug Detect (HPD) driver header file	432
pd_hal/pdss_hal.h	CCG PD PHY driver module header file	438
scb/i2c.h	I2C slave driver header file	511
system/boot.h	Bootloader support header file	518
system/ccgx_host_sdk_desc.h	CCGx Host SDK API Reference Guide Introduction	525
system/ccgx_version.h	This file defines the base firmware stack version for CCGx	525
system/flash.h	Flash command handler header file	526
system/gpio.h	GPIO and IO mapping control functions	532
system/instrumentation.h	Header file containing application instrumentation definitions	544
system/srom.h	SROM Code header file	545
system/status.h	API return status definitions	546
system/system.h	Support functions and definitions for bootloader and flash updates	547
system/timer.h	Soft timer header file	554
system/timer_id.h	Soft timer identifier definitions	560
system/utils.h	General utility macros and definitions for CCGx firmware stack	566
ucsi/ucsi.h	USB Type-C Connector System Software Interface (UCSI) header file	576

Chapter 5

Data Structure Documentation

5.1 pd_do_t::ACT_CBL_VDO Struct Reference

```
#include <pd.h>
```

Data Fields

- uint32_t [usb_ss_sup](#): 3
- uint32_t [sop_dp](#): 1
- uint32_t [vbus_thru_cbl](#): 1
- uint32_t [vbus_cur](#): 2
- uint32_t [rsvd1](#): 2
- uint32_t [max_vbus_volt](#): 2
- uint32_t [cbl_term](#): 2
- uint32_t [cbl_latency](#): 4
- uint32_t [typec_plug](#): 1
- uint32_t [typec_abc](#): 2
- uint32_t [rsvd2](#): 1
- uint32_t [vdo_version](#): 3
- uint32_t [cbl_fw_ver](#): 4
- uint32_t [cbl_hw_ver](#): 4

5.1.1 Detailed Description

Active cable VDO structure as defined by PD 3.0.

5.1.2 Field Documentation

5.1.2.1 cbl_fw_ver

```
uint32_t cbl_fw_ver
```

Cable firmware version.

5.1.2.2 cbl_hw_ver`uint32_t cbl_hw_ver`

Cable hardware version.

5.1.2.3 cbl_latency`uint32_t cbl_latency`

Cable latency.

5.1.2.4 cbl_term`uint32_t cbl_term`

Cable termination and VConn power requirement.

5.1.2.5 max_vbus_volt`uint32_t max_vbus_volt`

Max. VBus voltage supported.

5.1.2.6 rsvd1`uint32_t rsvd1`

Reserved field.

5.1.2.7 rsvd2`uint32_t rsvd2`

Reserved field.

5.1.2.8 sop_dp`uint32_t sop_dp`

Whether SOP" controller is present.

5.1.2.9 typec_abc`uint32_t typec_abc`

Cable plug type.

5.1.2.10 typec_plug`uint32_t typec_plug`

Reserved field.

5.1.2.11 usb_ss_sup

uint32_t usb_ss_sup

USB signalling supported by the cable.

5.1.2.12 vbus_cur

uint32_t vbus_cur

VBus current supported by the cable.

5.1.2.13 vbus_thru_cbl

uint32_t vbus_thru_cbl

Whether cable conducts VBus through.

5.1.2.14 vdo_version

uint32_t vdo_version

VDO version.

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

- [pd_common/pd.h](#)

5.2 pd_do_t::ACT_CBL_VDO_1 Struct Reference

```
#include <pd.h>
```

Data Fields

- uint32_t [usb_ss_sup](#): 3
- uint32_t [sop_dp](#): 1
- uint32_t [vbus_thru_cbl](#): 1
- uint32_t [vbus_cur](#): 2
- uint32_t [sbu_type](#): 1
- uint32_t [sbu_supp](#): 1
- uint32_t [max_vbus_volt](#): 2
- uint32_t [cbl_term](#): 2
- uint32_t [cbl_latency](#): 4
- uint32_t [rsvd1](#): 1
- uint32_t [typec_abc](#): 2
- uint32_t [rsvd2](#): 1
- uint32_t [vdo_version](#): 3
- uint32_t [cbl_fw_ver](#): 4
- uint32_t [cbl_hw_ver](#): 4

5.2.1 Detailed Description

Active Cable VDO 1 structure as defined by PD 3.0, Version 1.2.

5.2.2 Field Documentation

5.2.2.1 cbl_fw_ver

uint32_t cbl_fw_ver

Cable firmware version.

5.2.2.2 cbl_hw_ver

uint32_t cbl_hw_ver

Cable hardware version.

5.2.2.3 cbl_latency

uint32_t cbl_latency

Cable latency.

5.2.2.4 cbl_term

uint32_t cbl_term

Cable termination and VConn power requirement.

5.2.2.5 max_vbus_volt

uint32_t max_vbus_volt

Max. VBus voltage supported.

5.2.2.6 rsvd1

uint32_t rsvd1

Reserved field.

5.2.2.7 rsvd2

uint32_t rsvd2

Reserved field.

5.2.2.8 sbu_supp

uint32_t sbu_supp

Whether SBU connections are supported, 1=Not supported.

5.2.2.9 sbu_type

uint32_t sbu_type

Whether SBU connections are passive/active.

5.2.2.10 sop_dp

uint32_t sop_dp

Whether SOP" controller is present.

5.2.2.11 typec_abc

uint32_t typec_abc

Cable plug type.

5.2.2.12 usb_ss_sup

uint32_t usb_ss_sup

USB signalling supported by the cable.

5.2.2.13 vbus_cur

uint32_t vbus_cur

VBus current supported by the cable.

5.2.2.14 vbus_thru_cbl

uint32_t vbus_thru_cbl

Whether cable conducts VBus through.

5.2.2.15 vdo_version

uint32_t vdo_version

VDO version.

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

- [pd_common/pd.h](#)

5.3 pd_do_t::ACT_CBL_VDO_2 Struct Reference

```
#include <pd.h>
```

Data Fields

- uint32_t [usb_gen](#): 1
- uint32_t [rsvd0](#): 1
- uint32_t [opt_isolated](#): 1

- uint32_t `ss_lanes`: 1
- uint32_t `ss_supp`: 1
- uint32_t `usb2_supp`: 1
- uint32_t `usb2_hub_hops`: 2
- uint32_t `usb4_supp`: 1
- uint32_t `active_el`: 1
- uint32_t `phy_conn`: 1
- uint32_t `u3_u0_trans`: 1
- uint32_t `u3_power`: 3
- uint32_t `rsvd1`: 1
- uint32_t `shutdown_temp`: 8
- uint32_t `max_op_temp`: 8

5.3.1 Detailed Description

Active Cable VDO 2 structure as defined by PD 3.0, Version 1.2.

5.3.2 Field Documentation

5.3.2.1 `active_el`

uint32_t `active_el`

Active element.

5.3.2.2 `max_op_temp`

uint32_t `max_op_temp`

Maximum operating temperature.

5.3.2.3 `opt_isolated`

uint32_t `opt_isolated`

Optically Isolated Active Cable.

5.3.2.4 `phy_conn`

uint32_t `phy_conn`

Physical connection.

5.3.2.5 `rsvd0`

uint32_t `rsvd0`

Reserved field.

5.3.2.6 rsvd1

uint32_t rsvd1

Reserved field.

5.3.2.7 shutdown_temp

uint32_t shutdown_temp

Shutdown temperature.

5.3.2.8 ss_lanes

uint32_t ss_lanes

Whether cable supports 1 or 2 USB 3.2 lanes.

5.3.2.9 ss_supp

uint32_t ss_supp

Whether cable supports USB 3.2 signaling.

5.3.2.10 u3_power

uint32_t u3_power

USB 3.2 U3 power.

5.3.2.11 u3_u0_trans

uint32_t u3_u0_trans

Type of USB U3 to U0 transition.

5.3.2.12 usb2_hub_hops

uint32_t usb2_hub_hops

Number of USB 2.0 hub hops contributed by the cable.

5.3.2.13 usb2_supp

uint32_t usb2_supp

Whether cable supports USB 2.0 data.

5.3.2.14 usb4_supp

uint32_t usb4_supp

Whether cable supports USB 4.

5.3.2.15 usb_gen

uint32_t usb_gen

USB Generation.

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

- [pd_common/pd.h](#)

5.4 pd_do_t::ADO_ALERT Struct Reference

```
#include <pd.h>
```

Data Fields

- uint32_t [rsvd1](#):16
- uint32_t [hot_swap_bats](#):4
- uint32_t [fixed_bats](#):4
- uint32_t [rsvd2](#):1
- uint32_t [bat_status_change](#):1
- uint32_t [ocp](#):1
- uint32_t [otp](#):1
- uint32_t [op_cond_change](#):1
- uint32_t [src_input_change](#):1
- uint32_t [ovp](#):1

5.4.1 Detailed Description

PD 3.0 Alert Data Object.

5.4.2 Field Documentation

5.4.2.1 bat_status_change

uint32_t bat_status_change

Battery status changed.

5.4.2.2 fixed_bats

uint32_t fixed_bats

Identifies fixed batteries whose status has changed.

5.4.2.3 hot_swap_bats

uint32_t hot_swap_bats

Identifies hot-swappable batteries whose status has changed.

5.4.2.4 ocp

uint32_t ocp

Over-Current event status.

5.4.2.5 op_cond_change

uint32_t op_cond_change

Operating conditions changed.

5.4.2.6 otp

uint32_t otp

Over-Temperature event status.

5.4.2.7 ovp

uint32_t ovp

Over-Voltage event status.

5.4.2.8 rsvd1

uint32_t rsvd1

Reserved field.

5.4.2.9 rsvd2

uint32_t rsvd2

Reserved field.

5.4.2.10 src_input_change

uint32_t src_input_change

Power source input changed.

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

- [pd_common/pd.h](#)

5.5 alt_cfg_settings_t Struct Reference

```
#include <pd.h>
```

Data Fields

- uint16_t [table_len](#)
- uint8_t [dfp_mask](#)
- uint8_t [ufp_mask](#)

5.5.1 Detailed Description

Struct to hold the Alt modes settings.

5.5.2 Field Documentation

5.5.2.1 dfp_mask

```
uint8_t dfp_mask
```

DFP alt modes mask

5.5.2.2 table_len

```
uint16_t table_len
```

Table length in bytes

5.5.2.3 ufp_mask

```
uint8_t ufp_mask
```

UFP alt modes mask

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

- [pd_common/pd.h](#)

5.6 alt_mode_evt_t::ALT_MODE_EVT Struct Reference

```
#include <alt_modes_mngr.h>
```

Data Fields

- uint32_t [data_role](#): 1
- uint32_t [alt_mode_evt](#): 7
- uint32_t [alt_mode](#): 8
- uint32_t [svid](#): 16

5.6.1 Detailed Description

Struct containing alternate modes manager event/command .

5.6.2 Field Documentation

5.6.2.1 alt_mode

```
uint32_t alt_mode
```

Alt mode ID.

5.6.2.2 alt_mode_evt

```
uint32_t alt_mode_evt
```

Alt mode event index from alt_mode_app_evt_t structure.

5.6.2.3 data_role

```
uint32_t data_role
```

Current event sender data role.

5.6.2.4 svid

```
uint32_t svid
```

Alt mode related SVID.

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

- [app/alt_mode/alt_modes_mngr.h](#)

5.7 alt_mode_evt_t::ALT_MODE_EVT_DATA Struct Reference

```
#include <alt_modes_mngr.h>
```

Data Fields

- uint32_t [evt_data](#): 24
- uint32_t [evt_type](#): 8

5.7.1 Detailed Description

Struct containing alternate modes manager event/command data.

5.7.2 Field Documentation

5.7.2.1 evt_data

```
uint32_t evt_data
```

Alt mode event's data.

5.7.2.2 evt_type

```
uint32_t evt_type
```

Alt mode event's type.

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

- [app/alt_mode/alt_modes_mngr.h](#)

5.8 alt_mode_evt_t Union Reference

```
#include <alt_modes_mngr.h>
```

Data Structures

- struct [ALT_MODE_EVT](#)
- struct [ALT_MODE_EVT_DATA](#)

Data Fields

- [uint32_t val](#)
- struct [alt_mode_evt_t::ALT_MODE_EVT alt_mode_event](#)
- struct [alt_mode_evt_t::ALT_MODE_EVT_DATA alt_mode_event_data](#)

5.8.1 Detailed Description

Alt modes manager application event/command structure.

5.8.2 Field Documentation

5.8.2.1 alt_mode_event

```
struct alt_mode_evt_t::ALT_MODE_EVT alt_mode_event
```

Union containing the alt mode event value.

5.8.2.2 alt_mode_event_data

```
struct alt_mode_evt_t::ALT_MODE_EVT_DATA alt_mode_event_data
```

Union containing the alt mode event's data value.

5.8.2.3 val

```
uint32_t val
```

Integer field used for direct manipulation of reason code.

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

- [app/alt_mode/alt_modes_mngr.h](#)

5.9 alt_mode_hw_evt_t::ALT_MODE_HW_EVT Struct Reference

```
#include <alt_mode_hw.h>
```

Data Fields

- uint32_t [evt_data](#): 16
- uint32_t [hw_type](#): 8
- uint32_t [data_role](#): 8

5.9.1 Detailed Description

Struct containing alternate modes HW event/command .

5.9.2 Field Documentation

5.9.2.1 data_role

```
uint32_t data_role
```

Current data role.

5.9.2.2 evt_data

```
uint32_t evt_data
```

Current event/command data.

5.9.2.3 hw_type

```
uint32_t hw_type
```

HW type event/command related to.

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

- [app/alt_mode/alt_mode_hw.h](#)

5.10 alt_mode_hw_evt_t Union Reference

```
#include <alt_mode_hw.h>
```

Data Structures

- struct [ALT_MODE_HW_EVT](#)

Data Fields

- uint32_t [val](#)
- struct [alt_mode_hw_evt_t::ALT_MODE_HW_EVT](#) [hw_evt](#)

5.10.1 Detailed Description

Alt mode HW application event/command structure.

5.10.2 Field Documentation

5.10.2.1 hw_evt

```
struct alt_mode_hw_evt_t::ALT_MODE_HW_EVT hw_evt
```

Union containing the application HW event/command value.

5.10.2.2 val

```
uint32_t val
```

Integer field used for direct manipulation of reason code.

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

- [app/alt_mode/alt_mode_hw.h](#)

5.11 alt_mode_info_t Struct Reference

```
#include <alt_modes_mngr.h>
```

Data Fields

- [alt_mode_state_t mode_state](#)
- [alt_mode_state_t sop_state \[SOP_DPRIME+1\]](#)
- [uint8_t vdo_max_numb](#)
- [uint8_t obj_pos](#)
- [uint8_t cbl_obj_pos](#)
- [uint8_t alt_mode_id](#)
- [pd_do_t vdm_header](#)
- [pd_do_t * vdo \[SOP_DPRIME+1\]](#)
- [uint8_t vdo_numb \[SOP_DPRIME+1\]](#)
- [alt_mode_cbk_t cbk](#)
- [bool is_active](#)
- [bool custom_att_obj_pos](#)
- [bool uvdm_supp](#)
- [bool set_mux_isolate](#)
- [bool app_evt_needed](#)
- [alt_mode_evt_t app_evt_data](#)
- [alt_mode_app_cbk_t eval_app_cmd](#)

5.11.1 Detailed Description

This structure holds all necessary information for interaction between alt modes manager and selected alternate mode.

5.11.2 Field Documentation

5.11.2.1 alt_mode_id

uint8_t alt_mode_id

Alternate mode ID.

5.11.2.2 app_evt_data

alt_mode_evt_t app_evt_data

APP event data.

5.11.2.3 app_evt_needed

bool app_evt_needed

APP event flag.

5.11.2.4 cbk

alt_mode_cbk_t cbk

Alternate mode callback function.

5.11.2.5 cbl_obj_pos

uint8_t cbl_obj_pos

Cabel object position.

5.11.2.6 custom_att_obj_pos

bool custom_att_obj_pos

Object position field in Att VDM used by alt mode as custom.

5.11.2.7 eval_app_cmd

alt_mode_app_cbk_t eval_app_cmd

APP command cbk.

5.11.2.8 is_active

bool is_active

Active mode flag.

5.11.2.9 mode_state

`alt_mode_state_t` mode_state

Alternate mode state.

5.11.2.10 obj_pos

`uint8_t` obj_pos

Alternate mode object position.

5.11.2.11 set_mux_isolate

`bool` set_mux_isolate

Flag to indicate if MUX should be set to safe state while ENTER/EXIT alt mode is being processed.

5.11.2.12 sop_state

`alt_mode_state_t` sop_state[SOP_DPRIME+1]

VDM state for SOP/SOP'/SOP" packets.

5.11.2.13 uvdm_supp

`bool` uvdm_supp

Flag to indicate if alt mode support unstructured VDMs.

5.11.2.14 vdm_header

`pd_do_t` vdm_header

Buffer to hold VDM header.

5.11.2.15 vdo

`pd_do_t*` vdo[SOP_DPRIME+1]

Pointers array to alt mode VDO buffers

5.11.2.16 vdo_max_numb

`uint8_t` vdo_max_numb

Maximum number of VDO that alt mode can handle

5.11.2.17 vdo_numb

`uint8_t` vdo_numb[SOP_DPRIME+1]

Current number of VDOs used for processing in VDO buffers

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

- [app/alt_mode/alt_modes_mgr.h](#)

5.12 alt_mode_reg_info_t Struct Reference

```
#include <alt_modes_mngr.h>
```

Data Fields

- [uint8_t atch_type](#)
- [uint8_t data_role](#)
- [uint8_t alt_mode_id](#)
- [sop_t cbl_sop_flag](#)
- [pd_do_t svid_emca_vdo](#)
- [pd_do_t svid_vdo](#)
- [const atch_tgt_info_t * atch_tgt_info](#)
- [alt_mode_app_evt_t app_evt](#)

5.12.1 Detailed Description

This structure holds all necessary information on Discovery Mode stage for supported alternate mode when alt modes manager registers new alt mode.

5.12.2 Field Documentation

5.12.2.1 alt_mode_id

```
uint8_t alt_mode_id
```

Alt mode ID.

5.12.2.2 app_evt

```
alt_mode_app_evt_t app_evt
```

APP event.

5.12.2.3 atch_tgt_info

```
const atch_tgt_info_t* atch_tgt_info
```

Attached trgets info (dev/ama/cbl) from Discovery ID response.

5.12.2.4 atch_type

```
uint8_t atch_type
```

Target of disc svid (cable or device/ama).

5.12.2.5 cbl_sop_flag

```
sop_t cbl_sop_flag
```

Sop indication flag.

5.12.2.6 data_role

`uint8_t data_role`

Current data role.

5.12.2.7 svid_emca_vdo

`pd_do_t svid_emca_vdo`

SVID VDO from cable Discovery mode response.

5.12.2.8 svid_vdo

`pd_do_t svid_vdo`

SVID VDO from Discovery mode SOP response.

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

- [app/alt_mode/alt_modes_mgr.h](#)

5.13 app_cbk_t Struct Reference

```
#include <pd.h>
```

Data Fields

- `void(* app_event_handler)(uint8_t port, app_evt_t evt, const void *dat)`
- `void(* psrc_set_voltage)(uint8_t port, uint16_t volt_mV)`
- `void(* psrc_set_current)(uint8_t port, uint16_t cur_10mA)`
- `void(* psrc_enable)(uint8_t port, pwr_ready_cbk_t pwr_ready_handler)`
- `void(* psrc_disable)(uint8_t port, pwr_ready_cbk_t pwr_ready_handler)`
- `bool(* vconn_enable)(uint8_t port, uint8_t channel)`
- `void(* vconn_disable)(uint8_t port, uint8_t channel)`
- `bool(* vconn_is_present)(uint8_t port)`
- `bool(* vbus_is_present)(uint8_t port, uint16_t volt, int8_t per)`
- `void(* vbus_discharge_on)(uint8_t port)`
- `void(* vbus_discharge_off)(uint8_t port)`
- `void(* psnk_set_voltage)(uint8_t port, uint16_t volt_mV)`
- `void(* psnk_set_current)(uint8_t port, uint16_t cur_10mA)`
- `void(* psnk_enable)(uint8_t port)`
- `void(* psnk_disable)(uint8_t port, sink_discharge_off_cbk_t snk_discharge_off_handler)`
- `void(* eval_src_cap)(uint8_t port, const pd_packet_t *src_cap, app_resp_cbk_t app_resp_handler)`
- `void(* eval_rdo)(uint8_t port, pd_do_t rdo, app_resp_cbk_t app_resp_handler)`
- `void(* eval_dr_swap)(uint8_t port, app_resp_cbk_t app_resp_handler)`
- `void(* eval_pr_swap)(uint8_t port, app_resp_cbk_t app_resp_handler)`
- `void(* eval_vconn_swap)(uint8_t port, app_resp_cbk_t app_resp_handler)`
- `void(* eval_vdm)(uint8_t port, const pd_packet_t *vdm, vdm_resp_cbk_t vdm_resp_handler)`
- `void(* eval_fr_swap)(uint8_t port, app_resp_cbk_t app_resp_handler)`
- `uint16_t(* vbus_get_value)(uint8_t port)`
- `uint32_t(* psrc_get_voltage)(uint8_t port)`
- `void(* eval_enter_usb)(uint8_t port, const pd_packet_t *eudo, app_resp_cbk_t app_resp_handler)`

5.13.1 Detailed Description

Struct to hold the application interface. The application is expected to fill the structure with pointers to functions that use the on-board circuitry to accomplish tasks like source/sink power turn on/off. All the functions in this structure must be non-blocking and take minimum execution time.

Warning

The application must check the callback pointer passed by the stack is not NULL.

5.13.2 Field Documentation

5.13.2.1 app_event_handler

```
void(* app_event_handler) (uint8_t port, app_evt_t evt, const void *dat)
```

App event handler callback.

5.13.2.2 eval_dr_swap

```
void(* eval_dr_swap) (uint8_t port, app_resp_cbk_t app_resp_handler)
```

Handles DR swap request received by port.

5.13.2.3 eval_enter_usb

```
void(* eval_enter_usb) (uint8_t port, const pd_packet_t *eudo, app_resp_cbk_t app_resp_handler)
```

Function to evaluate Enter USB request.

5.13.2.4 eval_fr_swap

```
void(* eval_fr_swap) (uint8_t port, app_resp_cbk_t app_resp_handler)
```

Handle FR swap request received by the specified port.

5.13.2.5 eval_pr_swap

```
void(* eval_pr_swap) (uint8_t port, app_resp_cbk_t app_resp_handler)
```

Handles pr swap request received by port.

5.13.2.6 eval_rdo

```
void(* eval_rdo) (uint8_t port, pd_do_t rdo, app_resp_cbk_t app_resp_handler)
```

Evaluate sink request message.

5.13.2.7 eval_src_cap

```
void(* eval_src_cap) (uint8_t port, const pd_packet_t *src_cap, app_resp_cbk_t app_resp_handler)
```

Evaluate received source caps and provide the RDO to be used to negotiate contract.

5.13.2.8 eval_vconn_swap

```
void(* eval_vconn_swap) (uint8_t port, app_resp_cbk_t app_resp_handler)
```

Handles VCONN swap request received by port.

5.13.2.9 eval_vdm

```
void(* eval_vdm) (uint8_t port, const pd_packet_t *vdm, vdm_resp_cbk_t vdm_resp_handler)
```

Handle VDMs (all structured/unstructured VDMs need to be handled) received by the port.

5.13.2.10 psnk_disable

```
void(* psnk_disable) (uint8_t port, sink_discharge_off_cbk_t snk_discharge_off_handler)
```

Disable power sink related circuitry.

5.13.2.11 psnk_enable

```
void(* psnk_enable) (uint8_t port)
```

Enable power sink related circuitry.

5.13.2.12 psnk_set_current

```
void(* psnk_set_current) (uint8_t port, uint16_t cur_10mA)
```

Set power sink current, in 10mA units.

5.13.2.13 psnk_set_voltage

```
void(* psnk_set_voltage) (uint8_t port, uint16_t volt_mV)
```

Set power sink voltage, in mV units.

5.13.2.14 psrc_disable

```
void(* psrc_disable) (uint8_t port, pwr_ready_cbk_t pwr_ready_handler)
```

Disable the power supply. The pwr_ready_handler, if not NULL, must be called when the power supply has been discharged to Vsafe0V.

5.13.2.15 psrc_enable

```
void(* psrc_enable) (uint8_t port, pwr_ready_cbk_t pwr_ready_handler)
```

Enable the power supply. The pwr_ready_handler, if not NULL, must be called when the power supply is ready.

5.13.2.16 psrc_get_voltage

```
uint32_t(* psrc_get_voltage) (uint8_t port)
```

Get expected VBUS value in mV from application. This is to include any additional compensation done for drops.

5.13.2.17 psrc_set_current

```
void(* psrc_set_current) (uint8_t port, uint16_t cur_10mA)
```

Set power source current in 10mA units.

5.13.2.18 psrc_set_voltage

```
void(* psrc_set_voltage) (uint8_t port, uint16_t volt_mV)
```

Set power source voltage in mV units.

5.13.2.19 vbus_discharge_off

```
void(* vbus_discharge_off) (uint8_t port)
```

Turn off VBUS discharge circuit.

5.13.2.20 vbus_discharge_on

```
void(* vbus_discharge_on) (uint8_t port)
```

Turn on VBUS discharge circuit.

5.13.2.21 vbus_get_value

```
uint16_t(* vbus_get_value) (uint8_t port)
```

Get current VBUS value in mV from application.

5.13.2.22 vbus_is_present

```
bool(* vbus_is_present) (uint8_t port, uint16_t volt, int8_t per)
```

Check whether VBus voltage is within expected range.

5.13.2.23 vconn_disable

```
void(* vconn_disable) (uint8_t port, uint8_t channel)
```

Turn VCONN supply OFF.

5.13.2.24 vconn_enable

```
bool(* vconn_enable) (uint8_t port, uint8_t channel)
```

Turn VCONN supply ON. Return true if VCONN was turned ON.

5.13.2.25 vconn_is_present

```
bool(* vconn_is_present) (uint8_t port)
```

Check whether VConn supply is ON.

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

- [pd_common/pd.h](#)

5.14 app_resp_t Struct Reference

```
#include <pd.h>
```

Data Fields

- [pd_do_t resp_do](#)
- [app_req_status_t req_status](#)

5.14.1 Detailed Description

Struct to hold response to policy manager. Note: The [app_resp_t](#) structure is only used for responses that have a single DO. This may need to get extended if more DOs are to be supported.

5.14.2 Field Documentation

5.14.2.1 req_status

```
app_req_status_t req_status
```

Request status.

5.14.2.2 resp_do

```
pd_do_t resp_do
```

Response data object.

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

- [pd_common/pd.h](#)

5.15 app_status_t Struct Reference

```
#include <app.h>
```

Data Fields

- [pwr_ready_cbk_t pwr_ready_cbk](#)
- [sink_discharge_off_cbk_t snk_dis_cbk](#)
- [app_resp_t app_resp](#)
- [vdm_resp_t vdm_resp](#)
- [uint16_t psrc_volt](#)
- [uint16_t psrc_volt_old](#)
- [uint16_t psnk_volt](#)
- [uint16_t psnk_cur](#)
- [uint8_t vdm_task_en](#)
- [uint8_t disc_cbl_pending](#)
- [uint8_t cbl_disc_id_finished](#)
- [uint8_t vdm_version](#)

- uint8_t alt_mode_trig_mask
- uint8_t dfp_alt_mode_mask
- uint8_t ufp_alt_mode_mask
- uint16_t custom_hpi_svid
- volatile uint8_t fault_status
- bool alt_mode_entered
- bool vdm_prcs_failed
- bool is_vbus_on
- bool is_vconn_on
- bool vdm_retry_pending
- bool psrc_rising
- bool cbl_rst_done
- bool trig_cbl_rst
- bool cur_fb_enabled
- bool ld_sw_ctrl
- bool bc_12_src_disabled
- bool is_mux_busy
- vdm_resp_cbk_t vdm_resp_cbk
- bool is_vdm_pending
- mux_poll_fnc_cbk_t mux_poll_cbk
- bool keep_vconn_src
- uint8_t app_pending_swaps
- uint8_t actv_swap_type
- uint8_t actv_swap_count
- uint16_t actv_swap_delay
- uint8_t turn_on_temp_limit
- uint8_t turn_off_temp_limit
- bool is_hot_shutdown
- bool usb4_active
- uint8_t usb4_data_rst_cnt
- bool debug_acc_attached
- bool retimer_dis_req
- bool usb2_supp
- bool usb3_supp
- bool skip_mux_config
- bool cable_retimer_supp

5.15.1 Detailed Description

This structure hold all variables related to application layer functionality.

5.15.2 Field Documentation

5.15.2.1 actv_swap_count

```
uint8_t actv_swap_count
```

Denotes number of active swap attempts completed.

5.15.2.2 `actv_swap_delay`

`uint16_t actv_swap_delay`

Delay to be applied between repeated swap attempts.

5.15.2.3 `actv_swap_type`

`uint8_t actv_swap_type`

Denotes the active Swap operation.

5.15.2.4 `alt_mode_entered`

`bool alt_mode_entered`

Flag to indicate is alternate modes currently entered.

5.15.2.5 `alt_mode_trig_mask`

`uint8_t alt_mode_trig_mask`

Mask to indicate which alt mode should be enabled by EC.

5.15.2.6 `app_pending_swaps`

`uint8_t app_pending_swaps`

Variable denoting the types of swap operation that are pending.

5.15.2.7 `app_resp`

`app_resp_t app_resp`

Buffer for APP responses.

5.15.2.8 `bc_12_src_disabled`

`bool bc_12_src_disabled`

BC 1.2 source disabled flag.

5.15.2.9 `cable_retimer_supp`

`bool cable_retimer_supp`

Retimer supported flag for Ridge related applications.

5.15.2.10 `cbl_disc_id_finished`

`uint8_t cbl_disc_id_finished`

Flag to indicate that cable disc id finished.

5.15.2.11 cbl_rst_done

```
bool cbl_rst_done
```

Flag to indicate that cable reset was provided.

5.15.2.12 cur_fb_enabled

```
bool cur_fb_enabled
```

Indicates that current foldback is enabled

5.15.2.13 custom_hpi_svid

```
uint16_t custom_hpi_svid
```

Holds custom alternate mode SVID received from HPI.

5.15.2.14 debug_acc_attached

```
bool debug_acc_attached
```

Debug accessory attach status

5.15.2.15 dfp_alt_mode_mask

```
uint8_t dfp_alt_mode_mask
```

Mask to enable DFP alternate modes.

5.15.2.16 disc_cbl_pending

```
uint8_t disc_cbl_pending
```

Flag to indicate is cable discovery is pending.

5.15.2.17 fault_status

```
volatile uint8_t fault_status
```

Fault status bits for this port.

5.15.2.18 is_hot_shutdown

```
bool is_hot_shutdown
```

Indicates that hot shutdown detected

5.15.2.19 is_mux_busy

```
bool is_mux_busy
```

Flag to indicate that mux is switching.

5.15.2.20 is_vbus_on

bool is_vbus_on

Is supplying VBUS flag.

5.15.2.21 is_vconn_on

bool is_vconn_on

Is supplying VCONN flag.

5.15.2.22 is_vdm_pending

bool is_vdm_pending

VDM handling flag for MUX callback.

5.15.2.23 keep_vconn_src

bool keep_vconn_src

Flag indicating that we should keep VConn source role.

5.15.2.24 ld_sw_ctrl

bool ld_sw_ctrl

Indicates whether the VBUS load switch control is active or not.

5.15.2.25 mux_poll_cbk

[mux_poll_fnc_cbk_t](#) mux_poll_cbk

Holds pointer to MUX polling function.

5.15.2.26 psnk_cur

uint16_t psnk_cur

Current PSink current in 10mA units.

5.15.2.27 psnk_volt

uint16_t psnk_volt

Current PSink voltage in mV units.

5.15.2.28 psrc_rising

bool psrc_rising

Voltage ramp up/down.

5.15.2.29 psrc_volt`uint16_t psrc_volt`

Current Psource voltage in mV

5.15.2.30 psrc_volt_old`uint16_t psrc_volt_old`

Old Psource voltage in mV

5.15.2.31 pwr_ready_cbk`pwr_ready_cbk_t pwr_ready_cbk`

Registered Power source callback.

5.15.2.32 retimer_dis_req`bool retimer_dis_req`

Flag to indicate disable Retimer request in ridge layer

5.15.2.33 skip_mux_config`bool skip_mux_config`

Flag to indicate do not configure MUX

5.15.2.34 snk_dis_cbk`sink_discharge_off_cbk_t snk_dis_cbk`

Registered Power sink callback.

5.15.2.35 trig_cbl_rst`bool trig_cbl_rst`

Flag to trigger cable reset.

5.15.2.36 turn_off_temp_limit`uint8_t turn_off_temp_limit`

Temperature threshold to turn off internal FET

5.15.2.37 turn_on_temp_limit`uint8_t turn_on_temp_limit`

Temperature threshold to turn on internal FET after turn off condition

5.15.2.38 ufp_alt_mode_mask

```
uint8_t ufp_alt_mode_mask
```

Mask to enable UFP alternate modes.

5.15.2.39 usb2_supp

```
bool usb2_supp
```

USB2 supported flag for Ridge related applications.

5.15.2.40 usb3_supp

```
bool usb3_supp
```

USB3 supported flag for Ridge related applications.

5.15.2.41 usb4_active

```
bool usb4_active
```

Indicates that USB4 mode was entered

5.15.2.42 usb4_data_rst_cnt

```
uint8_t usb4_data_rst_cnt
```

Indicates number of Dat Reset retries

5.15.2.43 vdm_prcs_failed

```
bool vdm_prcs_failed
```

Flag to indicate is vdm process failed.

5.15.2.44 vdm_resp

```
vdm_resp_t vdm_resp
```

Buffer for VDM responses.

5.15.2.45 vdm_resp_cbk

```
vdm_resp_cbk_t vdm_resp_cbk
```

VDM response handler callback.

5.15.2.46 vdm_retry_pending

```
bool vdm_retry_pending
```

Whether VDM retry on timeout is pending.

5.15.2.47 vdm_task_en

uint8_t vdm_task_en

Flag to indicate is vdm task manager enabled.

5.15.2.48 vdm_version

uint8_t vdm_version

Live VDM version.

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

- [app/app.h](#)

5.16 atch_tgt_info_t Struct Reference

```
#include <vdm_task_mngr.h>
```

Data Fields

- [pd_do_t tgt_id_header](#)
- [pd_do_t ama_vdo](#)
- const [pd_do_t](#) * [cbl_vdo](#)
- uint16_t [tgt_svid](#) [[MAX_SVID_VDO_SUPP](#)]
- uint16_t [cbl_svid](#) [[MAX_SVID_VDO_SUPP](#)]

5.16.1 Detailed Description

This struct holds alternate mode discovery information which is used by alt modes manager.

5.16.2 Field Documentation

5.16.2.1 ama_vdo

[pd_do_t](#) ama_vdo

Holds AMA discovery ID response VDO .

5.16.2.2 cbl_svid

uint16_t cbl_svid [[MAX_SVID_VDO_SUPP](#)]

Holds received SVID for cable.

5.16.2.3 cbl_vdo

const [pd_do_t](#)* cbl_vdo

Pointer to cable VDO.

5.16.2.4 tgt_id_header

`pd_do_t` `tgt_id_header`

Holds Device/AMA discovery ID header .

5.16.2.5 tgt_svid

`uint16_t` `tgt_svid`[`MAX_SVID_VDO_SUPP`]

Holds received SVID for Device/AMA.

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

- `app/alt_mode/vdm_task_mgr.h`

5.17 auto_cfg_settings_t Struct Reference

```
#include <pd.h>
```

Data Fields

- `uint8_t` `table_len`
- `bool` `policy_mgr_en`
- `uint8_t` `sys_pwr`
- `uint8_t` `port_pwr`
- `uint8_t` `reserved_0` [2]
- `bool` `pps_en`
- `bool` `unconstrained_pwr_en`
- `uint8_t` `vin_throttling_ctrl`
- `uint8_t` `vin_oc1`
- `uint8_t` `vin_oc2`
- `uint8_t` `vin_oc3`
- `sensor_data_t` `sensor_data` [4]

5.17.1 Detailed Description

Struct to hold the automotive charger settings.

5.17.2 Field Documentation

5.17.2.1 policy_mgr_en

`bool` `policy_mgr_en`

Source policy manager Enable/Disable

5.17.2.2 port_pwr

`uint8_t` `port_pwr`

VBUS per port power in Watts

5.17.2.3 pps_en

bool pps_en

PPS Enable/Disable

5.17.2.4 reserved_0

uint8_t reserved_0[2]

Reserved for future use

5.17.2.5 sensor_data

sensor_data_t sensor_data[4]

Temperature throttling information for sensors

5.17.2.6 sys_pwr

uint8_t sys_pwr

VBUS system power in Watts

5.17.2.7 table_len

uint8_t table_len

Table length in bytes

5.17.2.8 unconstrained_pwr_en

bool unconstrained_pwr_en

Unconstrained Power Enable/Disable

5.17.2.9 vin_oc1

uint8_t vin_oc1

Minimum input voltage in 100mV units required for the system to supply OC1 (100%) power rating

5.17.2.10 vin_oc2

uint8_t vin_oc2

Minimum input voltage in 100mV units required for the system to supply OC2 (50%) power rating. To skip this power level, load with the 0xFF. To terminate at this level, load with 0.

5.17.2.11 vin_oc3

uint8_t vin_oc3

Minimum input voltage in 100mV units required for the system to supply OC3 (15W) power rating. To skip this power level, load with the 0xFF. To terminate at this level, load with 0. Beyond this threshold, the port shall be shutoff.

5.17.2.12 vin_throttling_ctrl

```
uint8_t vin_throttling_ctrl
```

Battery voltage throttling control

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

- [pd_common/pd.h](#)

5.18 bat_chg_params_t Struct Reference

```
#include <pd.h>
```

Data Fields

- [uint8_t table_len](#)
- [uint8_t reserved_0](#)
- [uint16_t vbatt_max_volt](#)
- [uint16_t vbatt_cutoff_volt](#)
- [uint16_t vbatt_dischg_en_volt](#)
- [uint16_t vbatt_max_cur](#)
- [uint16_t reserved_1](#)

5.18.1 Detailed Description

Struct to hold the battery charging parameters.

5.18.2 Field Documentation

5.18.2.1 reserved_0

```
uint8_t reserved_0
```

Reserved for future use

5.18.2.2 reserved_1

```
uint16_t reserved_1
```

Reserved for future use

5.18.2.3 table_len

```
uint8_t table_len
```

Battery charger parameter table length in bytes

5.18.2.4 vbatt_cutoff_volt

```
uint16_t vbatt_cutoff_volt
```

Minimum battery voltage below which device should stop discharging.

5.18.2.5 vbatt_dischg_en_volt

```
uint16_t vbatt_dischg_en_volt
```

Battery voltage at which discharge can be re-enabled.

5.18.2.6 vbatt_max_cur

```
uint16_t vbatt_max_cur
```

Maximum charging current allowed for the battery in mA.

5.18.2.7 vbatt_max_volt

```
uint16_t vbatt_max_volt
```

Maximum battery voltage in mV.

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

- [pd_common/pd.h](#)

5.19 pd_do_t::BAT_SNK Struct Reference

```
#include <pd.h>
```

Data Fields

- uint32_t [op_power](#): 10
- uint32_t [min_voltage](#): 10
- uint32_t [max_voltage](#): 10
- uint32_t [supply_type](#): 2

5.19.1 Detailed Description

Structure representing a Battery Supply PDO - Sink.

5.19.2 Field Documentation

5.19.2.1 max_voltage

```
uint32_t max_voltage
```

Maximum voltage in 50mV units.

5.19.2.2 min_voltage

uint32_t min_voltage

Minimum voltage in 50mV units.

5.19.2.3 op_power

uint32_t op_power

Maximum power in 250mW units.

5.19.2.4 supply_type

uint32_t supply_type

Supply type - should be 'b01'.

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

- [pd_common/pd.h](#)

5.20 pd_do_t::BAT_SRC Struct Reference

```
#include <pd.h>
```

Data Fields

- uint32_t [max_power](#): 10
- uint32_t [min_voltage](#): 10
- uint32_t [max_voltage](#): 10
- uint32_t [supply_type](#): 2

5.20.1 Detailed Description

Structure representing a Battery Supply PDO - Source.

5.20.2 Field Documentation

5.20.2.1 max_power

uint32_t max_power

Maximum power in 250mW units.

5.20.2.2 max_voltage

uint32_t max_voltage

Maximum voltage in 50mV units.

5.20.2.3 min_voltage

uint32_t min_voltage

Minimum voltage in 50mV units.

5.20.2.4 supply_type

uint32_t supply_type

Supply type - should be 'b01'.

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

- [pd_common/pd.h](#)

5.21 bb_settings_t Struct Reference

```
#include <pd.h>
```

Data Fields

- uint8_t [table_len](#)
- uint8_t volatile [bb_enable](#)
- uint8_t volatile [bb_always_on](#)
- uint8_t volatile [bb_option](#)
- uint16_t [bb_timeout](#)
- uint8_t volatile [bb_ec_present](#)
- uint8_t volatile [bb_bus_power](#)
- uint8_t volatile [bb_unique_container_id](#)
- uint8_t [reserved_1](#)
- uint16_t volatile [bb_bos_dscr_offset](#)
- uint16_t [bb_string_dscr_offset](#) [15]
- uint16_t [bb_vid](#)
- uint16_t [bb_pid](#)

5.21.1 Detailed Description

Struct to hold the Billboard settings.

5.21.2 Field Documentation

5.21.2.1 bb_always_on

uint8_t volatile [bb_always_on](#)

Byte 0x02: This field is valid only if the [bb_enable](#) is non-zero. The field determines when to enable the billboard interface. 0 => Enable the billboard device only on error. 1 => Always enable billboard on connection.

5.21.2.2 `bb_bos_dscr_offset`

```
uint16_t volatile bb_bos_dscr_offset
```

Byte 0x0A: This field is valid only for devices that have internal USB support. The field provides the offset inside the table where the BOS descriptor for the device is located. The BOS descriptor is mandatory for billboard support.

5.21.2.3 `bb_bus_power`

```
uint8_t volatile bb_bus_power
```

Byte 0x07: This field is valid only for devices that have internal USB support. The field indicates whether the device is bus powered or not.

5.21.2.4 `bb_ec_present`

```
uint8_t volatile bb_ec_present
```

Byte 0x06: EC present notification to external billboard controller.

5.21.2.5 `bb_enable`

```
uint8_t volatile bb_enable
```

Byte 0x01: This field indicates whether a billboard device is enabled. The usage and requirement for the billboard device is application specific. 0 => No billboard device. 1 => External billboard device. 2 => Internal billboard.

5.21.2.6 `bb_option`

```
uint8_t volatile bb_option
```

Byte 0x03: This field provides the various functionalities supported by the device. Bit 0 => 0 = No HID interface, 1 = Enable HID interface. Bits 1:7 => Reserved.

5.21.2.7 `bb_pid`

```
uint16_t bb_pid
```

Byte 0x2C: PID for the Billboard device.

5.21.2.8 `bb_string_dscr_offset`

```
uint16_t bb_string_dscr_offset[15]
```

Byte 0x0C: This field is valid only for devices that have internal USB support. The field provides the offset inside the table where the various string descriptors for the device are located. The descriptors are expected to be in a fixed order and is mandatory. The following are the usage models for the various string indices: 0 => Manufacturer string (mandatory). 1 => Product string (mandatory). 2 => Serial string (optional). 0000h = No serial string descriptor, FFFFh = Unique serial string generated by device, Any other offset is treated as a valid serial string. 3 => Configuration string (mandatory). 4 => Billboard interface string (mandatory). 5 => HID interface string (mandatory). 6 => Additional info URL string (mandatory). 7-8 => Alternate mode strings (mandatory for all valid modes).

5.21.2.9 `bb_timeout`

```
uint16_t bb_timeout
```

Byte 0x04: This field is valid only if `bb_enable` is non-zero. The field determines how long the billboard interface stays on, in seconds. FFFFh => Stays on until disconnect. 000Ah to FFFEh => Timeout in seconds.

5.21.2.10 bb_unique_container_id

```
uint8_t volatile bb_unique_container_id
```

Byte 0x08: This field is valid only for devices that have internal USB support. The field indicates whether the device creates the container ID descriptor or uses what is provided in the BOS descriptor.

5.21.2.11 bb_vid

```
uint16_t bb_vid
```

Byte 0x2A: VID for the Billboard device.

5.21.2.12 reserved_1

```
uint8_t reserved_1
```

Byte 0x09: Reserved area for future expansion.

5.21.2.13 table_len

```
uint8_t table_len
```

Byte 0x00: Billboard parameter table length in bytes.

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

- [pd_common/pd.h](#)

5.22 bc_dp_dm_state_t Union Reference

```
#include <battery_charging.h>
```

Data Fields

- `uint16_t state`
- `uint8_t d [2]`

5.22.1 Detailed Description

Union to hold Dp/Dm status.

5.22.2 Field Documentation

5.22.2.1 d

```
uint8_t d[2]
```

Individual status of Dp(d[0]) and Dm(d[1]).

5.22.2.2 state

```
uint16_t state
```

Combined status of Dp and Dm.

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

- [app/battery_charging.h](#)

5.23 bc_status_t Struct Reference

```
#include <battery_charging.h>
```

Data Fields

- [bc_charge_mode_t cur_mode](#)
- [bc_state_t bc_fsm_state](#)
- [uint32_t volatile bc_evt](#)
- [uint16_t cur_volt](#)
- [uint16_t cur_amp](#)
- [bool volatile connected](#)
- [bool volatile comp_rising](#)
- [bc_dp_dm_state_t volatile dp_dm_status](#)
- [bc_dp_dm_state_t volatile old_dp_dm_status](#)
- [bc_sink_timer_t cur_timer](#)
- [uint8_t afc_src_msg_count](#)
- [bool volatile afc_src_is_matched](#)
- [uint8_t afc_src_cur_match_byte](#)
- [uint8_t afc_src_last_match_byte](#)
- [uint8_t afc_src_matched_byte](#)
- [uint8_t afc_src_match_count](#)
- [uint8_t afc_tx_active](#)
- [bool attach](#)

5.23.1 Detailed Description

Struct to define battery charger status.

5.23.2 Field Documentation

5.23.2.1 afc_src_cur_match_byte

```
uint8_t afc_src_cur_match_byte
```

The currently matched AFC VI value.

5.23.2.2 afc_src_is_matched

```
bool volatile afc_src_is_matched
```

Status of VI match from the last received AFC byte.

5.23.2.3 `afc_src_last_match_byte`

```
uint8_t afc_src_last_match_byte
```

Previously matched AFC VI value.

5.23.2.4 `afc_src_match_count`

```
uint8_t afc_src_match_count
```

Number of AFC VI byte matches detected.

5.23.2.5 `afc_src_matched_byte`

```
uint8_t afc_src_matched_byte
```

Holds the VI byte that is matched in two out of the last three messages.

5.23.2.6 `afc_src_msg_count`

```
uint8_t afc_src_msg_count
```

Number of successful AFC message transfers. 3 transfers have to be successful before we can make any VI changes.

5.23.2.7 `afc_tx_active`

```
uint8_t afc_tx_active
```

Whether AFC message transmission is active.

5.23.2.8 `attach`

```
bool attach
```

Whether BC attach has been detected.

5.23.2.9 `bc_evt`

```
uint32_t volatile bc_evt
```

Bitmap representing event notifications to the state machine.

5.23.2.10 `bc_fsm_state`

```
bc_state_t bc_fsm_state
```

Current state of the BC state machine.

5.23.2.11 `comp_rising`

```
bool volatile comp_rising
```

Whether comparator is looking for a rising edge or falling edge.

5.23.2.12 `connected`

`bool volatile connected`

Whether BC connection is detected.

5.23.2.13 `cur_amp`

`uint16_t cur_amp`

Active supply current in 10 mA units.

5.23.2.14 `cur_mode`

`bc_charge_mode_t cur_mode`

Active charging scheme.

5.23.2.15 `cur_timer`

`bc_sink_timer_t cur_timer`

Identifies the BC timer that is running.

5.23.2.16 `cur_volt`

`uint16_t cur_volt`

Active VBus voltage in mV units.

5.23.2.17 `dp_dm_status`

`bc_dp_dm_state_t volatile dp_dm_status`

Debounced status of the DP/DM pins.

5.23.2.18 `old_dp_dm_status`

`bc_dp_dm_state_t volatile old_dp_dm_status`

Previous status of the DP/DM pins.

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

- [app/battery_charging.h](#)

5.24 `pd_do_t::BIST_DO` Struct Reference

```
#include <pd.h>
```

Data Fields

- `uint32_t rsvd1`: 16
- `uint32_t rsvd2`: 12
- `uint32_t mode`: 4

5.24.1 Detailed Description

Structure of a BIST data object.

5.24.2 Field Documentation

5.24.2.1 mode

```
uint32_t mode
```

BIST mode.

5.24.2.2 rsvd1

```
uint32_t rsvd1
```

Reserved field.

5.24.2.3 rsvd2

```
uint32_t rsvd2
```

Reserved field.

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

- [pd_common/pd.h](#)

5.25 cc_state_t Union Reference

```
#include <pd.h>
```

Data Fields

- [uint16_t state](#)
- [uint8_t cc \[2\]](#)

5.25.1 Detailed Description

Union to hold CC status.

5.25.2 Field Documentation

5.25.2.1 cc

```
uint8_t cc[2]
```

Individual status of CC1(cc[0]) and CC2(cc[1]).

5.25.2.2 state

uint16_t state

Combined status of CC1 and CC2.

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

- [pd_common/pd.h](#)

5.26 ccg_timer_t Struct Reference

```
#include <timer.h>
```

Data Fields

- [uint32_t count](#)
- [timer_cb_t cb](#)
- [uint16_t period](#)
- [timer_id_t id](#)

5.26.1 Detailed Description

Structure encapsulating information relating to a software timer.

This structure encapsulates all information related to a software timer object. The timer module maintains a list of these objects corresponding to the active software timers at any given point of time.

5.26.2 Field Documentation

5.26.2.1 cb

[timer_cb_t](#) cb

The callback function pointer for the timer object.

5.26.2.2 count

uint32_t count

The actual count value.

5.26.2.3 id

[timer_id_t](#) id

The ID handle associated with the timer object.

5.26.2.4 period

uint16_t period

Period of operation for the timer instance.

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

- [system/timer.h](#)

5.27 chg_cfg_params_t Struct Reference

```
#include <pd.h>
```

Data Fields

- [uint8_t table_len](#)
- [uint8_t src_sel](#)
- [uint8_t snk_sel](#)
- [uint8_t reserved_0](#)
- [uint8_t apple_src_id](#)
- [uint8_t qc_src_type](#)
- [uint8_t reserved_1](#)
- [uint8_t afc_src_cap_cnt](#)
- [uint8_t afc_src_caps](#) [16]

5.27.1 Detailed Description

Struct to hold the port legacy charging parameters.

5.27.2 Field Documentation

5.27.2.1 afc_src_cap_cnt

uint8_t afc_src_cap_cnt

Number of AFC source voltage-current capabilities supported.

5.27.2.2 afc_src_caps

uint8_t afc_src_caps[16]

AFC source voltage-current capabilities list.

5.27.2.3 apple_src_id

uint8_t apple_src_id

Apple charger brick ID to be used as source: Bit 0 -> 1 A Bit 1 -> 2.1 A Bit 2 -> 2.4 A

5.27.2.4 qc_src_type

```
uint8_t qc_src_type
```

Quick charge source type: Bit 0 -> QC 2.0 Class-A Bit 1 -> QC 2.0 Class-B Bit 2 -> QC 3.0 Class-A Bit 3 -> QC 3.0 Class-B

5.27.2.5 reserved_0

```
uint8_t reserved_0
```

Reserved for future use

5.27.2.6 reserved_1

```
uint8_t reserved_1
```

Reserved for future use

5.27.2.7 snk_sel

```
uint8_t snk_sel
```

Legacy charging sink selection bit mask: Bit 0 -> BC 1.2 Bit 1 -> Apple brick Other bits reserved

5.27.2.8 src_sel

```
uint8_t src_sel
```

Legacy charging source selection bit mask: Bit 0 -> BC 1.2 support Bit 1 -> Apple charger Bit 2 -> Quick charge Bit 3 -> AFC charger Other bits reserved

5.27.2.9 table_len

```
uint8_t table_len
```

Legacy charging parameter table length in bytes

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

- [pd_common/pd.h](#)

5.28 contract_t Struct Reference

```
#include <pd.h>
```

Data Fields

- [uint16_t cur_pwr](#)
- [uint16_t max_volt](#)
- [uint16_t min_volt](#)

5.28.1 Detailed Description

Structure to hold PD Contract information.

5.28.2 Field Documentation

5.28.2.1 cur_pwr

uint16_t cur_pwr

PD contract current/power.

5.28.2.2 max_volt

uint16_t max_volt

PD contract max voltage in mV

5.28.2.3 min_volt

uint16_t min_volt

PD contract min voltage in mV

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

- [pd_common/pd.h](#)

5.29 custom_alt_cfg_settings_t Struct Reference

```
#include <pd.h>
```

Data Fields

- [uint8_t table_len](#)
- [uint8_t reserved0](#)
- [uint16_t custom_alt_mode](#)
- [uint8_t custom_dfp_mask](#)
- [uint8_t custom_ufp_mask](#)
- [uint8_t reserved1 \[2\]](#)

5.29.1 Detailed Description

Struct to hold the Custom Alt mode settings.

5.29.2 Field Documentation

5.29.2.1 custom_alt_mode

uint16_t custom_alt_mode

Custom alt mode

5.29.2.2 custom_dfp_mask

```
uint8_t custom_dfp_mask
```

DFP alt modes mask

5.29.2.3 custom_ufp_mask

```
uint8_t custom_ufp_mask
```

UFP alt modes mask

5.29.2.4 reserved0

```
uint8_t reserved0
```

Reserved for future

5.29.2.5 reserved1

```
uint8_t reserved1[2]
```

Reserved for future

5.29.2.6 table_len

```
uint8_t table_len
```

Table length in bytes

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

- [pd_common/pd.h](#)

5.30 custom_host_cfg_settings_t Struct Reference

```
#include <pd.h>
```

Data Fields

- [uint8_t table_len](#)
- [uint8_t acc_mode_disable](#)
- [uint8_t rp_detach_disable](#)
- [uint8_t snk_path_enable](#)
- [uint32_t pwr_threshold](#)
- [uint8_t req_max_pwr](#)
- [uint8_t ext_powered_prs](#)
- [uint8_t pdo_sel_alg](#)
- [uint8_t reserved](#)

5.30.1 Detailed Description

Struct to hold the custom host settings.

5.30.2 Field Documentation

5.30.2.1 acc_mode_disable

uint8_t acc_mode_disable

Accessory mode enable/disable

5.30.2.2 ext_powered_prs

uint8_t ext_powered_prs

Option to accept PR_SWAP even if there is an external powered bit is set

5.30.2.3 pdo_sel_alg

uint8_t pdo_sel_alg

Source PDO selection algorithm (Default, max Power, Voltage or Current)

5.30.2.4 pwr_threshold

uint32_t pwr_threshold

Minimal power to turn the FET ON if source provides at least this.

5.30.2.5 req_max_pwr

uint8_t req_max_pwr

Option to request max current provided by the port partner instead of the current mentioned in the sink capabilities

5.30.2.6 reserved

uint8_t reserved

Reserved for future

5.30.2.7 rp_detach_disable

uint8_t rp_detach_disable

Option to enable/disable disconnect detect mechanism using Rp in Sink role

5.30.2.8 snk_path_enable

uint8_t snk_path_enable

Sink path enable/disable

5.30.2.9 table_len

uint8_t table_len

Table length in bytes

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

- [pd_common/pd.h](#)

5.31 pd_do_t::DFP_VDO Struct Reference

```
#include <pd.h>
```

Data Fields

- uint32_t [port_num](#): 5
- uint32_t [rsvd0](#): 19
- uint32_t [host_cap](#): 3
- uint32_t [rsvd1](#): 2
- uint32_t [vdo_version](#): 3

5.31.1 Detailed Description

DFP VDO.

5.31.2 Field Documentation

5.31.2.1 host_cap

uint32_t host_cap

Host capability.

5.31.2.2 port_num

uint32_t port_num

Unique port number to identify a specific port on a multi-port device.

5.31.2.3 rsvd0

uint32_t rsvd0

Reserved field.

5.31.2.4 rsvd1

uint32_t rsvd1

Reserved field.

5.31.2.5 vdo_version

uint32_t vdo_version

VDO version field.

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

- [pd_common/pd.h](#)

5.32 pd_do_t::DP_CONFIG_VDO Struct Reference

```
#include <pd.h>
```

Data Fields

- uint32_t [sel_conf](#): 2
- uint32_t [sign](#): 4
- uint32_t [rsvd1](#): 2
- uint32_t [dfp_asgmt](#): 8
- uint32_t [ufp_asgmt](#): 8
- uint32_t [rsvd2](#): 8

5.32.1 Detailed Description

DisplayPort configure VDO as defined by VESA spec.

5.32.2 Field Documentation

5.32.2.1 dfp_asgmt

uint32_t dfp_asgmt

DFP_D pin assignment.

5.32.2.2 rsvd1

uint32_t rsvd1

Reserved.

5.32.2.3 rsvd2

uint32_t rsvd2

Reserved field.

5.32.2.4 sel_conf

uint32_t sel_conf

Select configuration.

5.32.2.5 sign

uint32_t sign

Signalling for DP protocol.

5.32.2.6 ufp_asgmt

uint32_t ufp_asgmt

UFP_D pin assignment.

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

- [pd_common/pd.h](#)

5.33 pd_do_t::DP_STATUS_VDO Struct Reference

```
#include <pd.h>
```

Data Fields

- uint32_t [dfp_ufp_conn](#): 2
- uint32_t [pwr_low](#): 1
- uint32_t [en](#): 1
- uint32_t [mult_fun](#): 1
- uint32_t [usb_cfg](#): 1
- uint32_t [exit](#): 1
- uint32_t [hpd_state](#): 1
- uint32_t [hpd_irq](#): 1
- uint32_t [rsvd](#): 23

5.33.1 Detailed Description

DisplayPort status update VDO as defined by VESA spec.

5.33.2 Field Documentation

5.33.2.1 dfp_ufp_conn

uint32_t dfp_ufp_conn

Whether DFP_D/UFP_D is connected.

5.33.2.2 en

uint32_t en

DP functionality enabled.

5.33.2.3 exit

uint32_t exit

Exit DP mode request.

5.33.2.4 hpd_irq

uint32_t hpd_irq

HPD IRQ status.

5.33.2.5 hpd_state

uint32_t hpd_state

Current HPD state.

5.33.2.6 mult_fun

uint32_t mult_fun

Multi-function mode preferred.

5.33.2.7 pwr_low

uint32_t pwr_low

Low power mode.

5.33.2.8 rsvd

uint32_t rsvd

Reserved field.

5.33.2.9 usb_cfg

uint32_t usb_cfg

Request switch to USB configuration.

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

- [pd_common/pd.h](#)

5.34 dpm_pd_cmd_buf_t Struct Reference

```
#include <pd.h>
```

Data Fields

- [sop_t cmd_sop](#)
- [extd_msg_t extd_type](#)
- [pd_extd_hdr_t extd_hdr](#)

- `uint8_t no_of_cmd_do`
- `uint8_t * dat_ptr`
- `uint8_t timeout`
- `pd_do_t cmd_do [MAX_NO_OF_DO]`

5.34.1 Detailed Description

Struct to hold PD command buffer.

Warning

When providing pointer to the extended data make sure original buffer is always 4-byte aligned. i.e, even if 1 byte data is required, 4 bytes should be used to store that data.

5.34.2 Field Documentation

5.34.2.1 cmd_do

`pd_do_t cmd_do [MAX_NO_OF_DO]`

Command data objects.

5.34.2.2 cmd_sop

`sop_t cmd_sop`

SOP type

5.34.2.3 dat_ptr

`uint8_t* dat_ptr`

Data Pointer in case of extended message only

5.34.2.4 extd_hdr

`pd_extd_hdr_t extd_hdr`

Extended Header

5.34.2.5 extd_type

`extd_msg_t extd_type`

Extended Message Type

5.34.2.6 no_of_cmd_do

`uint8_t no_of_cmd_do`

No of data objects including VDM header

5.34.2.7 timeout

```
uint8_t timeout
```

Timeout value, in ms, for a response. If set to zero, the PD stack will not wait for a VDM response and jump to ready state after this buffer has been sent.

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

- [pd_common/pd.h](#)

5.35 dpm_status_t Struct Reference

```
#include <pd.h>
```

Data Fields

- [uint8_t port_role](#)
- [uint8_t dflt_port_role](#)
- [uint8_t src_cur_level](#)
- [uint8_t is_src_bat](#)
- [uint8_t is_snk_bat](#)
- [uint8_t snk_usb_susp_en](#)
- [uint8_t snk_usb_comm_en](#)
- [uint8_t swap_response](#)
- [uint8_t toggle](#)
- [uint8_t src_pdo_count](#)
- [uint8_t src_pdo_mask](#)
- [uint8_t snk_pdo_count](#)
- [uint8_t snk_pdo_mask](#)
- [uint8_t rp_supported](#)
- [uint8_t pd_support](#)
- [uint8_t try_src_snk](#)
- [uint8_t cbl_dsc](#)
- [uint8_t db_support](#)
- [uint8_t err_recov](#)
- [uint8_t port_disable](#)
- [uint8_t frs_enable](#)
- [uint8_t vconn_retain](#)
- [uint16_t reserved_3](#) [5]
- [pd_do_t src_pdo](#) [MAX_NO_OF_PDO]
- [pd_do_t snk_pdo](#) [MAX_NO_OF_PDO]
- [uint16_t snk_max_min](#) [MAX_NO_OF_PDO]
- [port_type_t cur_port_type](#)
- [port_role_t cur_port_role](#)
- [uint8_t volatile snk_cur_level](#)
- [uint8_t volatile bootup](#)
- [uint8_t volatile dead_bat](#)
- [uint8_t drp_period](#)
- [uint8_t src_period](#)
- [uint8_t snk_period](#)
- [uint8_t volatile skip_scan](#)
- [uint8_t polarity](#)
- [uint8_t rev_pol](#)

- uint8_t volatile connect
- bool volatile attach
- port_role_t role_at_connect
- pd_devtype_t attached_dev
- uint8_t volatile contract_exist
- bool volatile pd_connected
- uint8_t volatile pd_disabled
- uint8_t volatile ra_present
- bool volatile emca_present
- uint8_t volatile bist_cm2_enabled
- std_vdm_prod_t cbl_type
- std_vdm_ver_t cbl_vdm_version
- uint8_t volatile vconn_logical
- uint8_t cur_src_pdo_count
- uint8_t cur_snk_pdo_count
- uint8_t cbl_wait
- pe_cbl_state_t cbl_state
- uint8_t cbl_soft_reset_tried
- typec_fsm_state_t typec_fsm_state
- dpm_pd_cmd_t dpm_pd_cmd
- uint8_t dpm_pd_cmd_active
- uint8_t dpm_typec_cmd_active
- bool dpm_enabled
- bool dpm_init
- uint8_t dpm_safe_disable
- dpm_typec_cmd_t dpm_typec_cmd
- uint8_t src_cur_level_live
- cc_state_t cc_live
- cc_state_t cc_status
- cc_state_t cc_old_status
- cc_state_t cc_rd_status
- pd_rev_t spec_rev_sop_live
- pd_rev_t spec_rev_sop_prime_live
- pd_rev_t spec_rev_cbl
- pd_rev_t spec_rev_peer
- bool unchunk_sup_live
- bool unchunk_sup_peer
- bool snk_rp_detach_en
- bool cur_fb
- pd_ams_type non_intr_response
- bool fr_rx_disabled
- bool fr_rx_en_live
- bool fr_tx_disabled
- bool fr_tx_en_live
- volatile bool fault_active
- pe_fsm_state_t pe_fsm_state
- uint32_t volatile pe_evt
- uint32_t volatile typec_evt
- contract_t contract
- pd_do_t alert
- pd_do_t cbl_vdo
- bool cbl_mode_en
- uint16_t src_cap_start_delay
- app_cbk_t * app_cbk
- dpm_pd_cmd_cbk_t dpm_pd_cbk

- [dpm_typec_cmd_cbk_t](#) [dpm_typec_cbk](#)
- [dpm_pd_cmd_buf_t](#) * [cmd_p](#)
- [dpm_pd_cmd_buf_t](#) [dpm_cmd_buf](#)
- [uint16_t](#) [cur_snk_max_min](#) [[MAX_NO_OF_PDO](#)]
- [pd_do_t](#) [cur_src_pdo](#) [[MAX_NO_OF_PDO](#)]
- [pd_do_t](#) [cur_snk_pdo](#) [[MAX_NO_OF_PDO](#)]
- [pd_do_t](#) [src_cur_rdo](#)
- [pd_do_t](#) [src_last_rdo](#)
- [pd_do_t](#) [src_rdo](#)
- [pd_do_t](#) [snk_rdo](#)
- [pd_do_t](#) [snk_sel_pdo](#)
- [pd_do_t](#) [src_sel_pdo](#)
- [pd_packet_t](#) * [src_cap_p](#)
- [uint32_t](#) [padval](#)
- [pd_power_status_t](#) [port_status](#)
- [uint8_t](#) [ext_src_cap](#) [[CCG_PD_EXT_SRCCAP_SIZE](#)]
- [uint8_t](#) [pps_status](#) [[CCG_PD_EXT_PPS_STATUS_SIZE](#)]
- [uint8_t](#) [dpm_err_info](#)
- [bool](#) [pwr_limited](#)
- [pd_do_t](#) [cbl_vdo_2](#)
- [uint8_t](#) [ext_snk_cap](#) [[CCG_PD_EXT_SNKCAP_BUF_SIZE](#)]
- [uint32_t](#) [rand_base](#)
- [bool](#) [pd3_src_cc_busy](#)
- [uint8_t](#) [rev3_en](#)
- [uint8_t](#) [hw_drp_toggle_en](#)
- [uint8_t](#) [try_src_snk_dis](#)
- [uint8_t](#) [frs_rx_en](#)
- [uint8_t](#) [frs_tx_en](#)
- [uint8_t](#) [pps_src_en](#)
- [uint8_t](#) [usb4_en](#)

5.35.1 Detailed Description

PD Device Policy Status structure. This structure holds all of the configuration and status information associated with a port on the CCG device. Members of this structure should not be directly modified by any of the application code.

Warning

Initial elements of this structure maps directly to config table fields and hence must not be moved around or changed.

5.35.2 Field Documentation

5.35.2.1 alert

[pd_do_t](#) [alert](#)

Alert status

5.35.2.2 app_cbk

`app_cbk_t*` app_cbk

Application callback pointer.

5.35.2.3 attach

`bool` volatile attach

Port attached indication.

5.35.2.4 attached_dev

`pd_devtype_t` attached_dev

Type of device attached.

5.35.2.5 bist_cm2_enabled

`uint8_t` volatile bist_cm2_enabled

BIST Carrier Mode 2 going on

5.35.2.6 bootup

`uint8_t` volatile bootup

Flag to indicate chip bootup, used to check dead battery.

5.35.2.7 cbl_dsc

`uint8_t` cbl_dsc

Cable discovery control knob.

5.35.2.8 cbl_mode_en

`bool` cbl_mode_en

Whether cable supports alternate modes.

5.35.2.9 cbl_soft_reset_tried

`uint8_t` cbl_soft_reset_tried

Stores number of cable soft reset tries.

5.35.2.10 cbl_state

`pe_cbl_state_t` cbl_state

Cable discovery state machine state.

5.35.2.11 cbl_type

`std_vdm_prod_t` cbl_type

Stores the cable type.

5.35.2.12 cbl_vdm_version

`std_vdm_ver_t` cbl_vdm_version

Stores the cable VDM version.

5.35.2.13 cbl_vdo

`pd_do_t` cbl_vdo

Stores the last received cable VDO.

5.35.2.14 cbl_vdo_2

`pd_do_t` cbl_vdo_2

Stores the last received Active Cable VDO #2.

5.35.2.15 cbl_wait

`uint8_t` cbl_wait

Flag to indicate cable discovery is waiting for some event.

5.35.2.16 cc_live

`cc_state_t` cc_live

Live CC status.

5.35.2.17 cc_old_status

`cc_state_t` cc_old_status

Old CC status.

5.35.2.18 cc_rd_status

`cc_state_t` cc_rd_status

Rd status.

5.35.2.19 cc_status

`cc_state_t` cc_status

Current debounced CC status.

5.35.2.20 cmd_p

`dpm_pd_cmd_buf_t*` cmd_p

Pointer to DPM command buffer.

5.35.2.21 connect

`uint8_t` volatile connect

Port connected but not debounced yet.

5.35.2.22 contract

`contract_t` contract

Current pd contract.

5.35.2.23 contract_exist

`uint8_t` volatile contract_exist

PD contract exists indication.

5.35.2.24 cur_fb

`bool` cur_fb

Flag to indicate current foldback is active

5.35.2.25 cur_port_role

`port_role_t` cur_port_role

Current Port role: Sink or Source.

5.35.2.26 cur_port_type

`port_type_t` cur_port_type

Current port type: UFP or DFP.

5.35.2.27 cur_snk_max_min

`uint16_t` cur_snk_max_min[`MAX_NO_OF_PDO`]

Max min current/power of current sink capabilities.

5.35.2.28 cur_snk_pdo

`pd_do_t` cur_snk_pdo[`MAX_NO_OF_PDO`]

Current sink PDOs sent in sink cap messages.

5.35.2.29 cur_snk_pdo_count

uint8_t cur_snk_pdo_count

Sink PDO count in the last sent sink cap.

5.35.2.30 cur_src_pdo

pd_do_t cur_src_pdo[MAX_NO_OF_PDO]

Current source PDOs sent in source cap messages.

5.35.2.31 cur_src_pdo_count

uint8_t cur_src_pdo_count

Source PDO count in the last sent source cap.

5.35.2.32 db_support

uint8_t db_support

Dead battery support control knob.

5.35.2.33 dead_bat

uint8_t volatile dead_bat

Flag to indicate dead battery operation.

5.35.2.34 dflt_port_role

uint8_t dflt_port_role

Default port role: Sink or Source.

5.35.2.35 dpm_cmd_buf

dpm_pd_cmd_buf_t dpm_cmd_buf

Local DPM command buffer.

5.35.2.36 dpm_enabled

bool dpm_enabled

DPM enabled flag.

5.35.2.37 dpm_err_info

uint8_t dpm_err_info

Additional error status for DPM commands.

5.35.2.38 dpm_init

bool dpm_init

DPM Initialized flag.

5.35.2.39 dpm_pd_cbk

[dpm_pd_cmd_cbk_t](#) dpm_pd_cbk

Pointer to DPM PD callback function.

5.35.2.40 dpm_pd_cmd

[dpm_pd_cmd_t](#) dpm_pd_cmd

Current DPM PD command.

5.35.2.41 dpm_pd_cmd_active

uint8_t dpm_pd_cmd_active

Indicate DPM PD command was registered.

5.35.2.42 dpm_safe_disable

uint8_t dpm_safe_disable

DPM sage disable flag. Used to make sure that the port is disabled correctly after a fault occurrence.

5.35.2.43 dpm_typec_cbk

[dpm_typec_cmd_cbk_t](#) dpm_typec_cbk

Pointer to DPM Type C callback function.

5.35.2.44 dpm_typec_cmd

[dpm_typec_cmd_t](#) dpm_typec_cmd

Current DPM Type C command.

5.35.2.45 dpm_typec_cmd_active

uint8_t dpm_typec_cmd_active

Indicate DPM Type C command was registered.

5.35.2.46 drp_period

uint8_t drp_period

Time period for DRP toggling.

5.35.2.47 emca_present

```
bool volatile emca_present
```

EMCA cable present indication.

5.35.2.48 err_recov

```
uint8_t err_recov
```

Error recovery control knob.

5.35.2.49 ext_snk_cap

```
uint8_t ext_snk_cap[CCG_PD_EXT_SNKCAP_BUF_SIZE]
```

Buffer to hold extended sink caps.

5.35.2.50 ext_src_cap

```
uint8_t ext_src_cap[CCG_PD_EXT_SRCCAP_SIZE]
```

Buffer to hold extended source caps.

5.35.2.51 fault_active

```
volatile bool fault_active
```

Flag to indicate the a fault consition exists.

5.35.2.52 fr_rx_disabled

```
bool fr_rx_disabled
```

FRS receive disabled by EC.

5.35.2.53 fr_rx_en_live

```
bool fr_rx_en_live
```

Fast role signal detection enabled

5.35.2.54 fr_tx_disabled

```
bool fr_tx_disabled
```

FRS transmit disabled by EC.

5.35.2.55 fr_tx_en_live

```
bool fr_tx_en_live
```

Fast role signal auto send enabled

5.35.2.56 frs_enable

```
uint8_t frs_enable
```

FRS enable flags.

5.35.2.57 frs_rx_en

```
uint8_t frs_rx_en
```

Flag indicating that FRS as Initial Sink is supported.

5.35.2.58 frs_tx_en

```
uint8_t frs_tx_en
```

Flag indicating that FRS as Initial Source is supported.

5.35.2.59 hw_drp_toggle_en

```
uint8_t hw_drp_toggle_en
```

Flag indicating that Hardware based DRP toggling is enabled.

5.35.2.60 is_snk_bat

```
uint8_t is_snk_bat
```

Power sink is connected to a battery.

5.35.2.61 is_src_bat

```
uint8_t is_src_bat
```

Power source is connected to a battery.

5.35.2.62 non_intr_response

```
pd_ams_type non_intr_response
```

Flag to indicate stack is waiting for App response to a non interruptible AMS

5.35.2.63 padval

```
uint32_t padval
```

Fields below need to be properly aligned to 4 byte boundary

5.35.2.64 pd3_src_cc_busy

```
bool pd3_src_cc_busy
```

Whether we should keep Rp at SinkTxNG as a PD 3.0 Source.

5.35.2.65 pd_connected

bool volatile pd_connected

Ports are PD connected indication.

5.35.2.66 pd_disabled

uint8_t volatile pd_disabled

PD disabled indication.

5.35.2.67 pd_support

uint8_t pd_support

USB-PD supported.

5.35.2.68 pe_evt

uint32_t volatile pe_evt

Stores policy engine events.

5.35.2.69 pe_fsm_state

[pe_fsm_state_t](#) pe_fsm_state

Holds the current state of Policy Engine (PE).

5.35.2.70 polarity

uint8_t polarity

CC channel polarity (CC1=0, CC2=1)

5.35.2.71 port_disable

uint8_t port_disable

PD port disable flag.

5.35.2.72 port_role

uint8_t port_role

Port role: Sink, Source or Dual.

5.35.2.73 port_status

[pd_power_status_t](#) port_status

Port power status.

5.35.2.74 pps_src_en

```
uint8_t pps_src_en
```

Flag indicating that PPS source is supported.

5.35.2.75 pps_status

```
uint8_t pps_status[CCG_PD_EXT_PPS_STATUS_SIZE]
```

Buffer to hold PPS status.

5.35.2.76 pwr_limited

```
bool pwr_limited
```

Whether SRC PDOs have been limited due to cable capability.

5.35.2.77 ra_present

```
uint8_t volatile ra_present
```

Ra present indication.

5.35.2.78 rand_base

```
uint32_t rand_base
```

Temporary variable used for random number generation.

5.35.2.79 reserved_3

```
uint16_t reserved_3[5]
```

Reserved words for padding to 4-byte aligned address.

5.35.2.80 rev3_en

```
uint8_t rev3_en
```

Flag indicating that PD Revision 3.0 support is enabled.

5.35.2.81 rev_pol

```
uint8_t rev_pol
```

CC channel reverse polarity.

5.35.2.82 role_at_connect

```
port_role_t role_at_connect
```

Port role when the port moved to the attached state.

5.35.2.83 rp_supported

uint8_t rp_supported

Supported Rp values bit mask.

- Bit 0 => Default Rp supported.
- Bit 1 => 1.5 A Rp supported.
- Bit 2 => 3 A Rp supported.

5.35.2.84 skip_scan

uint8_t volatile skip_scan

Skip CC scanning control knob.

5.35.2.85 snk_cur_level

uint8_t volatile snk_cur_level

Type C current level in sink role.

5.35.2.86 snk_max_min

uint16_t snk_max_min[MAX_NO_OF_PDO]

Max min current from the config table or updated at runtime by the EC.

5.35.2.87 snk_pdo

pd_do_t snk_pdo[MAX_NO_OF_PDO]

Sink PDO loaded from the config table or updated at runtime by the EC.

5.35.2.88 snk_pdo_count

uint8_t snk_pdo_count

Sink PDO count from the config table or updated at runtime by the EC.

5.35.2.89 snk_pdo_mask

uint8_t snk_pdo_mask

Sink PDO mask from the config table or updated at runtime by the EC.

5.35.2.90 snk_period

uint8_t snk_period

Time period for which to stay as a SNK for a DRP device.

5.35.2.91 snk_rdo`pd_do_t snk_rdo`

Last RDO sent (when operating as a sink) that resulted in a contract.

5.35.2.92 snk_rp_detach_en`bool snk_rp_detach_en`

Flag to indicate sink will detach on Rp removal instead of VBUS removal.

5.35.2.93 snk_sel_pdo`pd_do_t snk_sel_pdo`

Selected PDO which resulted in contract (when sink).

5.35.2.94 snk_usb_comm_en`uint8_t snk_usb_comm_en`

USB communication supported indication.

5.35.2.95 snk_usb_susp_en`uint8_t snk_usb_susp_en`

USB suspend supported indication.

5.35.2.96 spec_rev_cbl`pd_rev_t spec_rev_cbl`

Spec revision of the currently connected cable.

5.35.2.97 spec_rev_peer`pd_rev_t spec_rev_peer`

Spec revision of the currently connected peer.

5.35.2.98 spec_rev_sop_live`pd_rev_t spec_rev_sop_live`

Current spec rev for SOP messages.

5.35.2.99 spec_rev_sop_prime_live`pd_rev_t spec_rev_sop_prime_live`

Current spec revision for SOP Prime/DPrime messages.

5.35.2.100 src_cap_p

`pd_packet_t*` src_cap_p

Pointer to the current/last source cap message received. May be NULL. Data pointed to by this should not be changed.

5.35.2.101 src_cap_start_delay

`uint16_t` src_cap_start_delay

Place holder for src cap start delay in milliseconds

5.35.2.102 src_cur_level

`uint8_t` src_cur_level

Type C current level in the source role.

5.35.2.103 src_cur_level_live

`uint8_t` src_cur_level_live

Current Type C current level in the source role.

5.35.2.104 src_cur_rdo

`pd_do_t` src_cur_rdo

Stores the current rdo received by source

5.35.2.105 src_last_rdo

`pd_do_t` src_last_rdo

Stores the last rdo received by source

5.35.2.106 src_pdo

`pd_do_t` src_pdo[MAX_NO_OF_PDO]

Source PDO loaded from the config table or updated at runtime by the EC.

5.35.2.107 src_pdo_count

`uint8_t` src_pdo_count

Source PDO count from the config table or updated at runtime by the EC.

5.35.2.108 src_pdo_mask

`uint8_t` src_pdo_mask

Source PDO mask from the config table or updated at runtime by the EC.

5.35.2.109 src_period`uint8_t src_period`

Time period for which to stay as a SRC for a DRP device.

5.35.2.110 src_rdo`pd_do_t src_rdo`

Last RDO received (when operating as a source) that resulted in a contract.

5.35.2.111 src_sel_pdo`pd_do_t src_sel_pdo`

Selected PDO which resulted in contract (when source).

5.35.2.112 swap_response`uint8_t swap_response`

Response to be sent for each USB-PD SWAP command.

- Bits 1:0 => DR_SWAP response.
- Bits 3:2 => PR_SWAP response.
- Bits 5:4 => VCONN_SWAP response. Allowed values are: 0=ACCEPT, 1=REJECT, 2=WAIT, 3=NOT_SUPPORTED.

5.35.2.113 toggle`uint8_t toggle`

DRP toggle is enabled.

5.35.2.114 try_src_snk`uint8_t try_src_snk`

Try Source/ Try Sink control knob.

5.35.2.115 try_src_snk_dis`uint8_t try_src_snk_dis`

Flag indicating that Try.SRC Try.SNK is disabled.

5.35.2.116 typec_evt`uint32_t volatile typec_evt`

Stores Type C events.

5.35.2.117 typec_fsm_state

`typec_fsm_state_t` typec_fsm_state

Type C generic FSM state.

5.35.2.118 unchunk_sup_live

`bool` unchunk_sup_live

Mutual unchunk support with the currently connected peer.

5.35.2.119 unchunk_sup_peer

`bool` unchunk_sup_peer

Unchunk support of the currently connected peer.

5.35.2.120 usb4_en

`uint8_t` usb4_en

Flag indicating that USB4 messages are supported.

5.35.2.121 vconn_logical

`uint8_t volatile` vconn_logical

VCONN logical status.

5.35.2.122 vconn_retain

`uint8_t` vconn_retain

Whether VConn should be retained in ON state.

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

- [pd_common/pd.h](#)

5.36 pd_do_t::ENTERUSB_VDO Struct Reference

```
#include <pd.h>
```

Data Fields

- `uint32_t` [rsvd0](#): 13
- `uint32_t` [host_present](#): 1
- `uint32_t` [host_tbt_supp](#): 1
- `uint32_t` [host_dp_supp](#): 1
- `uint32_t` [host_pcie_supp](#): 1
- `uint32_t` [cable_current](#): 2
- `uint32_t` [cable_type](#): 2
- `uint32_t` [cable_speed](#): 3
- `uint32_t` [rsvd1](#): 1

- uint32_t `usb3_drd`: 1
- uint32_t `usb4_drd`: 1
- uint32_t `rsvd2`: 1
- uint32_t `usb_mode`: 3
- uint32_t `rsvd3`: 1

5.36.1 Detailed Description

Enter USB Data Object.

5.36.2 Field Documentation

5.36.2.1 `cable_current`

uint32_t `cable_current`

Current carrying capacity of the cable.

5.36.2.2 `cable_speed`

uint32_t `cable_speed`

Data rate supported by the cable.

5.36.2.3 `cable_type`

uint32_t `cable_type`

Type of cable.

5.36.2.4 `host_dp_supp`

uint32_t `host_dp_supp`

Whether host supports DisplayPort.

5.36.2.5 `host_pcie_supp`

uint32_t `host_pcie_supp`

Whether host supports PCIe.

5.36.2.6 `host_present`

uint32_t `host_present`

Whether a host is connected.

5.36.2.7 `host_tbt_supp`

uint32_t `host_tbt_supp`

Whether host supports Thunderbolt.

5.36.2.8 rsvd0

uint32_t rsvd0

Reserved field.

5.36.2.9 rsvd1

uint32_t rsvd1

Reserved field.

5.36.2.10 rsvd2

uint32_t rsvd2

Reserved field.

5.36.2.11 rsvd3

uint32_t rsvd3

Reserved field.

5.36.2.12 usb3_drd

uint32_t usb3_drd

Whether the DFP is USB 3.2 DRD capable.

5.36.2.13 usb4_drd

uint32_t usb4_drd

Whether the DFP is USB4 DRD capable.

5.36.2.14 usb_mode

uint32_t usb_mode

Mode of USB communication (2.0, 3.2 or 4.0)

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

- [pd_common/pd.h](#)

5.37 pd_extd_hdr_t::EXTD_HDR_T Struct Reference

```
#include <pd.h>
```

Data Fields

- uint16_t [data_size](#): 9
- uint16_t [rsvd1](#): 1
- uint16_t [request](#): 1

- uint16_t `chunk_no`: 4
- uint16_t `chunked`: 1

5.37.1 Detailed Description

Extended header broken down into respective fields.

5.37.2 Field Documentation

5.37.2.1 `chunk_no`

uint16_t `chunk_no`

Bits 14:11 - Chunk number.

5.37.2.2 `chunked`

uint16_t `chunked`

Bit 15 - Chunked message.

5.37.2.3 `data_size`

uint16_t `data_size`

Bits 08:00 - Extended message size in bytes.

5.37.2.4 `request`

uint16_t `request`

Bit 10 - Chunk request.

5.37.2.5 `rsvd1`

uint16_t `rsvd1`

Bit 09 - Reserved.

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

- [pd_common/pd.h](#)

5.38 `pd_do_t::FIXED_SNK` Struct Reference

```
#include <pd.h>
```

Data Fields

- uint32_t `op_current`: 10
- uint32_t `voltage`: 10

- uint32_t `rsrvd`: 3
- uint32_t `fr_swap`: 2
- uint32_t `dr_swap`: 1
- uint32_t `usb_comm_cap`: 1
- uint32_t `ext_powered`: 1
- uint32_t `high_cap`: 1
- uint32_t `dual_role_power`: 1
- uint32_t `supply_type`: 2

5.38.1 Detailed Description

Structure representing a Fixed Supply PDO - Sink.

5.38.2 Field Documentation

5.38.2.1 `dr_swap`

uint32_t `dr_swap`

Data Role Swap supported.

5.38.2.2 `dual_role_power`

uint32_t `dual_role_power`

Dual role power support.

5.38.2.3 `ext_powered`

uint32_t `ext_powered`

Externally powered.

5.38.2.4 `fr_swap`

uint32_t `fr_swap`

FR swap support.

5.38.2.5 `high_cap`

uint32_t `high_cap`

Higher capability possible.

5.38.2.6 `op_current`

uint32_t `op_current`

Operational current in 10mA units.

5.38.2.7 rsrvd

uint32_t rsrvd

Reserved field.

5.38.2.8 supply_type

uint32_t supply_type

Supply type - should be 'b00.

5.38.2.9 usb_comm_cap

uint32_t usb_comm_cap

USB communication capability.

5.38.2.10 voltage

uint32_t voltage

Voltage in 50mV units.

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

- [pd_common/pd.h](#)

5.39 pd_do_t::FIXED_SRC Struct Reference

```
#include <pd.h>
```

Data Fields

- uint32_t [max_current](#): 10
- uint32_t [voltage](#): 10
- uint32_t [pk_current](#): 2
- uint32_t [reserved](#): 2
- uint32_t [unchunk_sup](#): 1
- uint32_t [dr_swap](#): 1
- uint32_t [usb_comm_cap](#): 1
- uint32_t [ext_powered](#): 1
- uint32_t [usb_suspend_sup](#): 1
- uint32_t [dual_role_power](#): 1
- uint32_t [supply_type](#): 2

5.39.1 Detailed Description

Structure representing a Fixed Supply PDO - Source.

5.39.2 Field Documentation

5.39.2.1 dr_swap

uint32_t dr_swap

Data Role Swap supported.

5.39.2.2 dual_role_power

uint32_t dual_role_power

Dual role power support.

5.39.2.3 ext_powered

uint32_t ext_powered

Externally powered.

5.39.2.4 max_current

uint32_t max_current

Maximum current in 100mA units.

5.39.2.5 pk_current

uint32_t pk_current

Peak current.

5.39.2.6 reserved

uint32_t reserved

Reserved field.

5.39.2.7 supply_type

uint32_t supply_type

Supply type - should be 'b00.

5.39.2.8 unchunk_sup

uint32_t unchunk_sup

Unchunked extended messages supported.

5.39.2.9 usb_comm_cap

uint32_t usb_comm_cap

USB communication capability.

5.39.2.10 usb_suspend_sup

uint32_t usb_suspend_sup

USB suspend supported.

5.39.2.11 voltage

uint32_t voltage

Voltage in 50mV units.

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

- [pd_common/pd.h](#)

5.40 fw_img_status_t Union Reference

```
#include <boot.h>
```

Data Structures

- struct [fw_mode_reason_t](#)

Data Fields

- uint8_t [val](#)
- struct [fw_img_status_t::fw_mode_reason_t](#) [status](#)

5.40.1 Detailed Description

Boot mode reason structure.

This structure holds status of FW images and boot mode request. If the CCGx device is running in Boot-loader mode, this register can be used to identify the reason for this. The register will report the validity of FW1 and FW2 binaries even in the case where the device is already running in FW1 or FW2 mode.

5.40.2 Field Documentation

5.40.2.1 status

```
struct fw_img_status_t::fw_mode_reason_t status
```

Struct containing the status fields in the boot mode reason value.

5.40.2.2 val

```
uint8_t val
```

Integer field used for direct manipulation of reason code.

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

- [system/boot.h](#)

5.41 fw_img_status_t::fw_mode_reason_t Struct Reference

```
#include <boot.h>
```

Data Fields

- [uint8_t boot_mode_request](#): 1
- [uint8_t reserved](#): 1
- [uint8_t fw1_invalid](#): 1
- [uint8_t fw2_invalid](#): 1
- [uint8_t reserved1](#): 4

5.41.1 Detailed Description

< Structure containing boot reason status bits.

5.41.2 Field Documentation

5.41.2.1 boot_mode_request

```
uint8_t boot_mode_request
```

Boot mode request made by FW.

5.41.2.2 fw1_invalid

```
uint8_t fw1_invalid
```

FW1 image invalid: 0=Valid, 1=Invalid.

5.41.2.3 fw2_invalid

```
uint8_t fw2_invalid
```

FW2 image invalid: 0=Valid, 1=Invalid.

5.41.2.4 reserved

```
uint8_t reserved
```

Reserved field: Will be zero.

5.41.2.5 reserved1

```
uint8_t reserved1
```

Reserved for later use.

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

- [system/boot.h](#)

5.42 i2c_scb_config_t Struct Reference

```
#include <i2c.h>
```

Data Fields

- [i2c_scb_mode_t mode](#)
- [uint8_t slave_address](#)
- [uint8_t slave_mask](#)
- [i2c_scb_clock_freq_t clock_freq](#)
- [i2c_scb_state_t i2c_state](#)
- [uint8_t * buffer](#)
- [uint16_t buf_size](#)
- [i2c_cb_fun_t cb_fun_ptr](#)
- [volatile uint16_t i2c_write_count](#)

5.42.1 Detailed Description

This structure holds all configuration associated with each I2C block.

This structure is used internally by the driver, and is not expected to be manipulated directly by the caller.

5.42.2 Field Documentation

5.42.2.1 buf_size

```
uint16_t buf_size
```

Size of receive buffer.

5.42.2.2 buffer

```
uint8_t* buffer
```

Buffer to be used to receive data.

5.42.2.3 cb_fun_ptr

```
i2c_cb_fun_t cb_fun_ptr
```

Callback function pointer.

5.42.2.4 clock_freq

```
i2c_scb_clock_freq_t clock_freq
```

Clock frequency. Only valid in master mode.

5.42.2.5 i2c_state

[i2c_scb_state_t](#) i2c_state

Current state of the I2C block.

5.42.2.6 i2c_write_count

volatile uint16_t i2c_write_count

Current index into receive buffer.

5.42.2.7 mode

[i2c_scb_mode_t](#) mode

I2C operating mode.

5.42.2.8 slave_address

uint8_t slave_address

Slave address. Only valid in slave modes.

5.42.2.9 slave_mask

uint8_t slave_mask

Slave address mask. Only valid in slave modes.

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

- [scb/i2c.h](#)

5.43 icl_tgl_cfg_settings_t Struct Reference

```
#include <pd.h>
```

Data Fields

- [uint8_t table_len](#)
- [uint8_t icl_i2c_pmc_address](#)
- [uint8_t icl_i2c_retimer_address](#) [2]
- [uint8_t icl_tgl_selection](#)
- [uint8_t icl_dual_retimer_enable](#)
- [uint16_t soc_mux_init_delay](#)
- [uint16_t soc_mux_config_delay](#)
- [uint8_t reserved0](#) [2]

5.43.1 Detailed Description

Struct to hold the ICL/TGL settings.

5.43.2 Field Documentation

5.43.2.1 icl_dual_retimer_enable

`uint8_t icl_dual_retimer_enable`

Retimer/Dual retimer enable/disable

5.43.2.2 icl_i2c_pmc_address

`uint8_t icl_i2c_pmc_address`

Configuring I2C slave address to Intel PMC

5.43.2.3 icl_i2c_retimer_address

`uint8_t icl_i2c_retimer_address[2]`

Configuring I2C master address to retimers

5.43.2.4 icl_tgl_selection

`uint8_t icl_tgl_selection`

Platform selection ICL/TGL

5.43.2.5 reserved0

`uint8_t reserved0[2]`

Reserved for future

5.43.2.6 soc_mux_config_delay

`uint16_t soc_mux_config_delay`

Soc Mux Config delay in milli seconds

5.43.2.7 soc_mux_init_delay

`uint16_t soc_mux_init_delay`

Soc Mux initialization delay in milli seconds

5.43.2.8 table_len

`uint8_t table_len`

Table length in bytes

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

- [pd_common/pd.h](#)

5.44 ocp_settings_t Struct Reference

```
#include <pd.h>
```

Data Fields

- [uint8_t table_len](#)
- [uint8_t threshold](#)
- [uint8_t debounce](#)
- [uint8_t retry_cnt](#)
- [uint8_t threshold2](#)
- [uint8_t debounce2](#)
- [uint8_t sense_res](#)
- [uint8_t cs_res](#)

5.44.1 Detailed Description

Struct to hold the OCP settings.

5.44.2 Field Documentation

5.44.2.1 cs_res

```
uint8_t cs_res
```

Current sense tuning resistor impedance in 100 Ohm units.

5.44.2.2 debounce

```
uint8_t debounce
```

OCP debounce period in milliseconds. OCP handling is only performed if the OCP condition persists for this duration.

5.44.2.3 debounce2

```
uint8_t debounce2
```

Debounce period in ms corresponding to secondary threshold.

5.44.2.4 retry_cnt

```
uint8_t retry_cnt
```

Number of consecutive OCP events allowed before the port is suspended by CCG firmware.

5.44.2.5 sense_res

```
uint8_t sense_res
```

Sense Resistor impedance in milli-Ohm units.

5.44.2.6 table_len

```
uint8_t table_len
```

Table length in bytes

5.44.2.7 threshold

```
uint8_t threshold
```

OCF threshold: Excess current in percentage of maximum allowed current.

5.44.2.8 threshold2

```
uint8_t threshold2
```

Secondary OCF threshold (corresponding to peak current condition). Set to 0 if not used.

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

- [pd_common/pd.h](#)

5.45 otp_settings_t Struct Reference

```
#include <pd.h>
```

Data Fields

- [uint8_t table_len](#)
- [uint8_t therm_type](#)
- [uint16_t cutoff_val](#)
- [uint16_t restart_val](#)
- [uint16_t debounce](#)
- [uint8_t therm_1_enable](#)
- [uint8_t therm_type_1](#)
- [uint16_t cutoff_val_1](#)
- [uint16_t restart_val_1](#)
- [uint8_t reserved_1](#) [2]

5.45.1 Detailed Description

Struct to hold the OTP settings.

5.45.2 Field Documentation

5.45.2.1 cutoff_val

```
uint16_t cutoff_val
```

Reading corresponding to the temperature at which OTP cutoff is to be performed (in mV or degrees C)

5.45.2.2 cutoff_val_1

uint16_t cutoff_val_1

Reading corresponding to the temperature at which OTP cutoff is to be performed (in mV or degrees C)

5.45.2.3 debounce

uint16_t debounce

OTP debounce duration in ms.

5.45.2.4 reserved_1

uint8_t reserved_1[2]

Reserved for future use

5.45.2.5 restart_val

uint16_t restart_val

Reading corresponding to the temperature at which system operation can be resumed (in mV or degrees C)

5.45.2.6 restart_val_1

uint16_t restart_val_1

Reading corresponding to the temperature at which system operation can be resumed (in mV or degrees C)

5.45.2.7 table_len

uint8_t table_len

Table length in bytes

5.45.2.8 therm_1_enable

uint8_t therm_1_enable

Enable/Disable the second thermistor

5.45.2.9 therm_type

uint8_t therm_type

Type of thermistor used for temperature sensing: 0 = Negative Temperature Coefficient (NTC) 1 = Positive Temperature Coefficient (PTC) 2 = Internal VBJT measurement.

5.45.2.10 therm_type_1

uint8_t therm_type_1

Type of thermistor used for temperature sensing: 0 = Negative Temperature Coefficient (NTC) 1 = Positive Temperature Coefficient (PTC)

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

- [pd_common/pd.h](#)

5.46 ovp_settings_t Struct Reference

```
#include <pd.h>
```

Data Fields

- [uint8_t table_len](#)
- [uint8_t threshold](#)
- [uint8_t debounce](#)
- [uint8_t retry_cnt](#)
- [uint8_t debounce_ms](#)

5.46.1 Detailed Description

Struct to hold the OVP settings.

5.46.2 Field Documentation

5.46.2.1 debounce

```
uint8_t debounce
```

OVP debounce duration in micro-seconds. If a non-zero debounce is specified, there can be an error of upto 35 us due to the device being in sleep mode.

5.46.2.2 debounce_ms

```
uint8_t debounce_ms
```

OVP debounce duration in mili-seconds. Accepted if `debounce == 0` and `debounce_ms != 0`

5.46.2.3 retry_cnt

```
uint8_t retry_cnt
```

Number of consecutive OVP events allowed before the port operation is suspended by CCG firmware.

5.46.2.4 table_len

```
uint8_t table_len
```

Table length in bytes

5.46.2.5 threshold

```
uint8_t threshold
```

OVP threshold: Excess voltage above expected value in percentage of expected voltage

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

- [pd_common/pd.h](#)

5.47 pd_do_t::PAS_CBL_VDO Struct Reference

```
#include <pd.h>
```

Data Fields

- [uint32_t usb_ss_sup](#): 3
- [uint32_t rsvd1](#): 2
- [uint32_t vbus_cur](#): 2
- [uint32_t rsvd2](#): 2
- [uint32_t max_vbus_volt](#): 2
- [uint32_t cbl_term](#): 2
- [uint32_t cbl_latency](#): 4
- [uint32_t typec_plug](#): 1
- [uint32_t typec_abc](#): 2
- [uint32_t rsvd3](#): 1
- [uint32_t vdo_version](#): 3
- [uint32_t cbl_fw_ver](#): 4
- [uint32_t cbl_hw_ver](#): 4

5.47.1 Detailed Description

Passive cable VDO structure as defined by PD 3.0.

5.47.2 Field Documentation

5.47.2.1 cbl_fw_ver

```
uint32_t cbl_fw_ver
```

Cable firmware version.

5.47.2.2 cbl_hw_ver

```
uint32_t cbl_hw_ver
```

Cable hardware version.

5.47.2.3 cbl_latency

```
uint32_t cbl_latency
```

Cable latency.

5.47.2.4 cbl_term`uint32_t cbl_term`

Cable termination and VConn power requirement.

5.47.2.5 max_vbus_volt`uint32_t max_vbus_volt`

Max. VBus voltage supported.

5.47.2.6 rsvd1`uint32_t rsvd1`

Reserved field.

5.47.2.7 rsvd2`uint32_t rsvd2`

Reserved field.

5.47.2.8 rsvd3`uint32_t rsvd3`

Reserved field.

5.47.2.9 typec_abc`uint32_t typec_abc`

Cable plug type.

5.47.2.10 typec_plug`uint32_t typec_plug`

Reserved field.

5.47.2.11 usb_ss_sup`uint32_t usb_ss_sup`

USB signalling supported by the cable.

5.47.2.12 vbus_cur`uint32_t vbus_cur`

VBus current supported by the cable.

5.47.2.13 vdo_version

```
uint32_t vdo_version
```

VDO version.

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

- [pd_common/pd.h](#)

5.48 pasc_valley_table_t Struct Reference

```
#include <pdss_hal.h>
```

Data Fields

- [uint16_t v0](#)
- [uint16_t v1](#)
- [uint16_t v2](#)
- [uint16_t max_cur](#)
- [uint8_t safe_valley](#)
- [uint8_t reserved_0](#) [3]
- [uint8_t table](#) [4][10]

5.48.1 Detailed Description

The structure contains the valley table to be used to implement the valley algorithm for PAG1S based power adapter solution. The table contains the frequency selection parameters at various voltage and load conditions.

Applicable devices: PAG1S.

5.48.2 Field Documentation

5.48.2.1 max_cur

```
uint16_t max_cur
```

Maximum current supported by this design in 10mA units. The 10 entries in the valley table corresponds to each 10% increment on this current rating.

5.48.2.2 reserved_0

```
uint8_t reserved_0[3]
```

Reserved byte to maintain alignment of the table.

5.48.2.3 safe_valley

```
uint8_t safe_valley
```

Safe (highest) valley threshold which can deliver maximum load without affecting performance. [1-64]. Loading invalid valley number can result in unstable / undefined behaviour.

5.48.2.4 table

```
uint8_t table[4][10]
```

Two dimensional valley number table which contains four entries matching each voltage thresholds. The last array contains any higher voltage above v2. $v0 \leq v1 \leq v2$. The 10 entries in each array corresponds to 10%, 20%.. load current. Valid valley number is between [1-64]. Parameter validation is not done and is expected to be loaded with valid values only. Valley number should be decrease as load increases.

5.48.2.5 v0

```
uint16_t v0
```

Voltage level V0 corresponding to first index in table.

5.48.2.6 v1

```
uint16_t v1
```

Voltage level V1 corresponding to second index in table. $V1 \geq V0$. Paramter validation is not done and is expected to be loaded with valid values only.

5.48.2.7 v2

```
uint16_t v2
```

Voltage level V2 corresponding to third index in table. $V2 \geq V1$. Paramter validation is not done and is expected to be loaded with valid values only.

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

- [pd_hal/pdss_hal.h](#)

5.49 pd_config_t Struct Reference

```
#include <pd.h>
```

Data Fields

- [uint16_t table_sign](#)
- [uint8_t table_type](#)
- [uint8_t application](#)
- [uint16_t table_version](#)
- [uint16_t table_size](#)
- [uint8_t table_checksum](#)
- [uint8_t flash_checksum](#)
- [uint16_t flashing_vid](#)
- [uint16_t flashing_mode](#)
- [uint8_t pd_port_cnt](#)
- [uint8_t reserved_0](#)
- [uint16_t mfg_info_offset](#)
- [volatile uint8_t mfg_info_length](#)
- [uint8_t cable_disc_cnt](#)
- [uint32_t db_event_mask](#)
- [uint16_t reserved_2](#) [4]
- [pd_port_config_t port_conf](#) [NO_OF_TYPEC_PORTS]

5.49.1 Detailed Description

Struct to hold the PD device configuration.

5.49.2 Field Documentation

5.49.2.1 application

uint8_t application

Byte 0x03: This field specifies the type of PD application supported: 0 => Notebook 1 => Tablet 2 => Passive Cable 3 => Active Cable 4 => Monitor 5 => Power Adapter 6 => Cable Adapter (Dongle)

5.49.2.2 cable_disc_cnt

uint8_t cable_disc_cnt

Byte 0x13: Number of cable discovery attempts to be made.

5.49.2.3 db_event_mask

uint32_t db_event_mask

Byte 0x14: Default event mask for reporting events in dead-battery scenario.

5.49.2.4 flash_checksum

uint8_t flash_checksum

Byte 0x09: One byte flash checksum. Used to ensure that binary sum of config table is 0.

5.49.2.5 flashing_mode

uint16_t flashing_mode

Byte 0x0C: Special Mode used for flashing in case of CC firmware updates.

5.49.2.6 flashing_vid

uint16_t flashing_vid

Byte 0x0A: Special VID used for flashing mode in case of CC firmware updates.

5.49.2.7 mfg_info_length

volatile uint8_t mfg_info_length

Byte 0x12: Length of manufacturer information in config table.

5.49.2.8 mfg_info_offset

uint16_t mfg_info_offset

Byte 0x10: Offset to manufacturer information in config table.

5.49.2.9 pd_port_cnt

uint8_t pd_port_cnt

Byte 0x0E: Number of PD ports enabled in the configuration.

5.49.2.10 port_conf

[pd_port_config_t](#) port_conf[NO_OF_TYPEC_PORTS]

Byte 0x20 (0x110): Configuration data for each USB-PD port.

5.49.2.11 reserved_0

uint8_t reserved_0

Byte 0x0F: Reserved byte for 2-byte alignment.

5.49.2.12 reserved_2

uint16_t reserved_2[4]

Byte 0x18: Reserved fields for future expansion.

5.49.2.13 table_checksum

uint8_t table_checksum

Byte 0x08: One byte configuration checksum. Calculated over bytes 10 to byte (size - 1).

5.49.2.14 table_sign

uint16_t table_sign

Byte 0x00: Two byte signature to indicate validity of the configuration.

5.49.2.15 table_size

uint16_t table_size

Byte 0x06: Size of the configuration table. Checksum is calculated over bytes 10 to size - 1.

5.49.2.16 table_type

uint8_t table_type

Byte 0x02: The table type indicates the type of solution. 1 => EMCA 2 => DFP 3 => UFP 4 => DRP

5.49.2.17 table_version

uint16_t table_version

Byte 0x04: Table version: This contains 4 bit major version, 4 bit minor version and 8 bit patch number.

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

- [pd_common/pd.h](#)

5.50 pd_contract_info_t Struct Reference

```
#include <pd.h>
```

Data Fields

- [pd_do_t rdo](#)
- [pd_contract_status_t status](#)

5.50.1 Detailed Description

Structure to hold PD Contract information passed with APP_EVT_PD_CONTRACT_NEGOTIATION_COMPLETE event to the application.

5.50.2 Field Documentation

5.50.2.1 rdo

[pd_do_t](#) rdo

RDO associated with the contract.

5.50.2.2 status

[pd_contract_status_t](#) status

Status of the contract.

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

- [pd_common/pd.h](#)

5.51 pd_do_t Union Reference

```
#include <pd.h>
```

Data Structures

- struct [ACT_CBL_VDO](#)
- struct [ACT_CBL_VDO_1](#)
- struct [ACT_CBL_VDO_2](#)
- struct [ADO_ALERT](#)
- struct [BAT_SNK](#)
- struct [BAT_SRC](#)
- struct [BIST_DO](#)
- struct [DFP_VDO](#)
- struct [DP_CONFIG_VDO](#)
- struct [DP_STATUS_VDO](#)
- struct [ENTERUSB_VDO](#)
- struct [FIXED_SNK](#)

- struct [FIXED_SRC](#)
- struct [PAS_CBL_VDO](#)
- struct [PPS_SNK](#)
- struct [PPS_SRC](#)
- struct [QC_4_0_DATA_VDO](#)
- struct [RDO_BAT](#)
- struct [RDO_BAT_GIVEBACK](#)
- struct [RDO_FIXED_VAR](#)
- struct [RDO_FIXED_VAR_GIVEBACK](#)
- struct [RDO_GEN](#)
- struct [RDO_GEN_GVB](#)
- struct [RDO_PPS](#)
- struct [SRC_GEN](#)
- struct [STD_AMA_VDO](#)
- struct [STD_AMA_VDO_PD3](#)
- struct [STD_CBL_VDO](#)
- struct [STD_CERT_VDO](#)
- struct [STD_DP_VDO](#)
- struct [STD_PROD_VDO](#)
- struct [STD_SVID_RESP_VDO](#)
- struct [STD_VDM_HDR](#)
- struct [STD_VDM_ID_HDR](#)
- struct [TBT_CBL_VDO](#)
- struct [TBT_UFP_VDO](#)
- struct [TBT_VDO](#)
- struct [UFP_VDO_1](#)
- struct [USTD_QC_4_0_HDR](#)
- struct [USTD_VDM_HDR](#)
- struct [VAR_SNK](#)
- struct [VAR_SRC](#)

Data Fields

- uint32_t val
- struct [pd_do_t::BIST_DO](#) bist_do
- struct [pd_do_t::FIXED_SRC](#) fixed_src
- struct [pd_do_t::VAR_SRC](#) var_src
- struct [pd_do_t::BAT_SRC](#) bat_src
- struct [pd_do_t::SRC_GEN](#) src_gen
- struct [pd_do_t::FIXED_SNK](#) fixed_snk
- struct [pd_do_t::VAR_SNK](#) var_snk
- struct [pd_do_t::BAT_SNK](#) bat_snk
- struct [pd_do_t::RDO_FIXED_VAR](#) rdo_fix_var
- struct [pd_do_t::RDO_FIXED_VAR_GIVEBACK](#) rdo_fix_var_gvb
- struct [pd_do_t::RDO_BAT](#) rdo_bat
- struct [pd_do_t::RDO_BAT_GIVEBACK](#) rdo_bat_gvb
- struct [pd_do_t::RDO_GEN](#) rdo_gen
- struct [pd_do_t::RDO_GEN_GVB](#) rdo_gen_gvb
- struct [pd_do_t::STD_VDM_HDR](#) std_vdm_hdr
- struct [pd_do_t::USTD_VDM_HDR](#) ustd_vdm_hdr
- struct [pd_do_t::USTD_QC_4_0_HDR](#) ustd_qc_4_0_hdr
- struct [pd_do_t::QC_4_0_DATA_VDO](#) qc_4_0_data_vdo
- struct [pd_do_t::STD_VDM_ID_HDR](#) std_id_hdr
- struct [pd_do_t::STD_CERT_VDO](#) std_cert_vdo

- struct `pd_do_t::STD_PROD_VDO` `std_prod_vdo`
- struct `pd_do_t::STD_CBL_VDO` `std_cbl_vdo`
- struct `pd_do_t::PAS_CBL_VDO` `pas_cbl_vdo`
- struct `pd_do_t::ACT_CBL_VDO` `act_cbl_vdo`
- struct `pd_do_t::ACT_CBL_VDO_1` `act_cbl_vdo1`
- struct `pd_do_t::ACT_CBL_VDO_2` `act_cbl_vdo2`
- struct `pd_do_t::STD_AMA_VDO` `std_ama_vdo`
- struct `pd_do_t::STD_AMA_VDO_PD3` `std_ama_vdo_pd3`
- struct `pd_do_t::STD_SVID_RESP_VDO` `std_svid_res`
- struct `pd_do_t::STD_DP_VDO` `std_dp_vdo`
- struct `pd_do_t::DP_STATUS_VDO` `dp_stat_vdo`
- struct `pd_do_t::DP_CONFIG_VDO` `dp_cfg_vdo`
- struct `pd_do_t::PPS_SRC` `pps_src`
- struct `pd_do_t::PPS_SNK` `pps_snk`
- struct `pd_do_t::RDO_PPS` `rdo_pps`
- struct `pd_do_t::ADO_ALERT` `ado_alert`
- struct `pd_do_t::TBT_UFP_VDO` `tbt_ufp_vdo`
- struct `pd_do_t::TBT_VDO` `tbt_vdo`
- struct `pd_do_t::TBT_CBL_VDO` `tbt_cbl_vdo`
- struct `pd_do_t::UFP_VDO_1` `ufp_vdo_1`
- struct `pd_do_t::DFP_VDO` `dfp_vdo`
- struct `pd_do_t::ENTERUSB_VDO` `enterusb_vdo`

5.51.1 Detailed Description

Union to hold a PD data object. All USB-PD data objects are 4-byte values which are interpreted according to the message type, length and object position. This union represents all possible interpretations of a USB-PD data object.

5.51.2 Field Documentation

5.51.2.1 `act_cbl_vdo`

```
struct pd_do_t::ACT_CBL_VDO act_cbl_vdo
```

DO interpreted as a PD 3.0 active cable VDO.

5.51.2.2 `act_cbl_vdo1`

```
struct pd_do_t::ACT_CBL_VDO_1 act_cbl_vdo1
```

DO interpreted as a PD 3.0 Active Cable VDO 1.

5.51.2.3 `act_cbl_vdo2`

```
struct pd_do_t::ACT_CBL_VDO_2 act_cbl_vdo2
```

DO interpreted as a PD 3.0 Active Cable VDO 2.

5.51.2.4 ado_alert

```
struct pd_do_t::ADO_ALERT ado_alert
```

DO interpreted as a PD 3.0 alert message.

5.51.2.5 bat_snk

```
struct pd_do_t::BAT_SNK bat_snk
```

DO interpreted as a Battery Supply PDO - Sink.

5.51.2.6 bat_src

```
struct pd_do_t::BAT_SRC bat_src
```

DO interpreted as a Battery Supply PDO - Source.

5.51.2.7 bist_do

```
struct pd_do_t::BIST_DO bist_do
```

DO interpreted as a BIST data object.

5.51.2.8 dfp_vdo

```
struct pd_do_t::DFP_VDO dfp_vdo
```

DO interpreted as UFP VDO1 data object.

5.51.2.9 dp_cfg_vdo

```
struct pd_do_t::DP_CONFIG_VDO dp_cfg_vdo
```

DO interpreted as a DisplayPort Configure command.

5.51.2.10 dp_stat_vdo

```
struct pd_do_t::DP_STATUS_VDO dp_stat_vdo
```

DO interpreted as a DisplayPort status update.

5.51.2.11 enterusb_vdo

```
struct pd_do_t::ENTERUSB_VDO enterusb_vdo
```

DO interpreted as an Enter USB Data Object.

5.51.2.12 fixed_snk

```
struct pd_do_t::FIXED_SNK fixed_snk
```

DO interpreted as a Fixed Supply PDO - Sink.

5.51.2.13 fixed_src

```
struct pd_do_t::FIXED_SRC fixed_src
```

DO interpreted as a Fixed Supply PDO - Source.

5.51.2.14 pas_cbl_vdo

```
struct pd_do_t::PAS_CBL_VDO pas_cbl_vdo
```

DO interpreted as a PD 3.0 passive cable VDO.

5.51.2.15 pps_snk

```
struct pd_do_t::PPS_SNK pps_snk
```

DO interpreted as a Programmable Power Supply - Sink.

5.51.2.16 pps_src

```
struct pd_do_t::PPS_SRC pps_src
```

DO interpreted as a Programmable Power Supply - Source.

5.51.2.17 qc_4_0_data_vdo

```
struct pd_do_t::QC_4_0_DATA_VDO qc_4_0_data_vdo
```

DO interpreted as a QC 4.0 Unstructured VDM data object.

5.51.2.18 rdo_bat

```
struct pd_do_t::RDO_BAT rdo_bat
```

DO interpreted as a Battery request.

5.51.2.19 rdo_bat_gvb

```
struct pd_do_t::RDO_BAT_GIVEBACK rdo_bat_gvb
```

DO interpreted as a Battery request with giveback.

5.51.2.20 rdo_fix_var

```
struct pd_do_t::RDO_FIXED_VAR rdo_fix_var
```

DO interpreted as a fixed/variable request.

5.51.2.21 rdo_fix_var_gvb

```
struct pd_do_t::RDO_FIXED_VAR_GIVEBACK rdo_fix_var_gvb
```

DO interpreted as a fixed/variable request with giveback.

5.51.2.22 rdo_gen

```
struct pd_do_t::RDO_GEN rdo_gen
```

DO interpreted as a generic request message.

5.51.2.23 rdo_gen_gvb

```
struct pd_do_t::RDO_GEN_GVB rdo_gen_gvb
```

DO interpreted as a generic request with giveback.

5.51.2.24 rdo_pps

```
struct pd_do_t::RDO_PPS rdo_pps
```

DO interpreted as a PPD Request.

5.51.2.25 src_gen

```
struct pd_do_t::SRC_GEN src_gen
```

DO interpreted as a generic PDO - Source.

5.51.2.26 std_ama_vdo

```
struct pd_do_t::STD_AMA_VDO std_ama_vdo
```

DO interpreted as a PD 2.0 AMA VDO.

5.51.2.27 std_ama_vdo_pd3

```
struct pd_do_t::STD_AMA_VDO_PD3 std_ama_vdo_pd3
```

DO interpreted as a PD 3.0 AMA VDO.

5.51.2.28 std_cbl_vdo

```
struct pd_do_t::STD_CBL_VDO std_cbl_vdo
```

DO interpreted as a PD 2.0 cable VDO.

5.51.2.29 std_cert_vdo

```
struct pd_do_t::STD_CERT_VDO std_cert_vdo
```

DO interpreted as a Cert Stat VDO.

5.51.2.30 std_dp_vdo

```
struct pd_do_t::STD_DP_VDO std_dp_vdo
```

DO interpreted as a DisplayPort Mode response.

5.51.2.31 std_id_hdr

```
struct pd_do_t::STD_VDM_ID_HDR std_id_hdr
```

DO interpreted as a Standard ID_HEADER VDO.

5.51.2.32 std_prod_vdo

```
struct pd_do_t::STD_PROD_VDO std_prod_vdo
```

DO interpreted as a Product VDO.

5.51.2.33 std_svid_res

```
struct pd_do_t::STD_SVID_RESP_VDO std_svid_res
```

DO interpreted as a DISCOVER_SVID response.

5.51.2.34 std_vdm_hdr

```
struct pd_do_t::STD_VDM_HDR std_vdm_hdr
```

DO interpreted as a Structured VDM header.

5.51.2.35 tbt_cbl_vdo

```
struct pd_do_t::TBT_CBL_VDO tbt_cbl_vdo
```

DO interpreted as a Thunderbolt Discovery response.

5.51.2.36 tbt_ufp_vdo

```
struct pd_do_t::TBT_UFP_VDO tbt_ufp_vdo
```

Data Object interpreted as a Thunderbolt3 mode VDO.

5.51.2.37 tbt_vdo

```
struct pd_do_t::TBT_VDO tbt_vdo
```

DO interpreted as a Thunderbolt Discovery response.

5.51.2.38 ufp_vdo_1

```
struct pd_do_t::UFP_VDO_1 ufp_vdo_1
```

DO interpreted as UFP VDO1 data object.

5.51.2.39 ustd_qc_4_0_hdr

```
struct pd_do_t::USTD_QC_4_0_HDR ustd_qc_4_0_hdr
```

DO interpreted as a QC 4.0 Unstructured VDM header.

5.51.2.40 `ustd_vdm_hdr`

```
struct pd_do_t::USTD_VDM_HDR ustd_vdm_hdr
```

DO interpreted as a Cypress unstructured VDM header.

5.51.2.41 `val`

```
uint32_t val
```

Data object interpreted as an unsigned integer value.

5.51.2.42 `var_snk`

```
struct pd_do_t::VAR_SNK var_snk
```

DO interpreted as a Variable Supply PDO - Sink.

5.51.2.43 `var_src`

```
struct pd_do_t::VAR_SRC var_src
```

DO interpreted as a Variable Supply PDO - Source.

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

- [pd_common/pd.h](#)

5.52 `pd_extd_hdr_t` Union Reference

```
#include <pd.h>
```

Data Structures

- struct [EXTD_HDR_T](#)

Data Fields

- `uint16_t val`
- struct `pd_extd_hdr_t::EXTD_HDR_T extd`

5.52.1 Detailed Description

Union to hold the PD extended header.

5.52.2 Field Documentation

5.52.2.1 `extd`

```
struct pd_extd_hdr_t::EXTD_HDR_T extd
```

Extended header broken down into respective fields.

5.52.2.2 val

uint16_t val

Extended header expressed as 2-byte integer value.

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

- [pd_common/pd.h](#)

5.53 pd_hdr_t::PD_HDR Struct Reference

```
#include <pd.h>
```

Data Fields

- uint32_t [msg_type](#): 5
- uint32_t [data_role](#): 1
- uint32_t [spec_rev](#): 2
- uint32_t [pwr_role](#): 1
- uint32_t [msg_id](#): 3
- uint32_t [len](#): 3
- uint32_t [extd](#): 1
- uint32_t [data_size](#): 9
- uint32_t [rsvd1](#): 1
- uint32_t [request](#): 1
- uint32_t [chunk_no](#): 4
- uint32_t [chunked](#): 1

5.53.1 Detailed Description

PD message header broken down into component fields. Includes 2-byte extended message header.

5.53.2 Field Documentation

5.53.2.1 chunk_no

uint32_t chunk_no

Bits 30:27 - Chunk number.

5.53.2.2 chunked

uint32_t chunked

Bit 31 - Chunked message.

5.53.2.3 data_role

uint32_t data_role

Bit 05 - Data role.

5.53.2.4 data_size

uint32_t data_size

Bits 24:16 - Extended message size in bytes.

5.53.2.5 extd

uint32_t extd

Bit 15 - Extended message.

5.53.2.6 len

uint32_t len

Bits 14:12 - Number of data objects.

5.53.2.7 msg_id

uint32_t msg_id

Bits 11:09 - Message ID.

5.53.2.8 msg_type

uint32_t msg_type

Bits 04:00 - Message type.

5.53.2.9 pwr_role

uint32_t pwr_role

Bit 08 - Power role.

5.53.2.10 request

uint32_t request

Bit 26 - Chunk request.

5.53.2.11 rsvd1

uint32_t rsvd1

Bit 25 - Reserved.

5.53.2.12 spec_rev

uint32_t spec_rev

Bits 07:06 - Spec revision.

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

- [pd_common/pd.h](#)

5.54 pd_hdr_t Union Reference

```
#include <pd.h>
```

Data Structures

- struct [PD_HDR](#)

Data Fields

- [uint32_t val](#)
- struct [pd_hdr_t::PD_HDR](#) [hdr](#)

5.54.1 Detailed Description

Union to hold the PD header defined by the USB-PD specification. Lower 16 bits hold the message header and the upper 16 bits hold the extended message header (where applicable).

5.54.2 Field Documentation

5.54.2.1 [hdr](#)

```
struct pd\_hdr\_t::PD\_HDR hdr
```

PD message header split into component fields.

5.54.2.2 [val](#)

```
uint32_t val
```

Header expressed as a 32-bit word.

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

- [pd_common/pd.h](#)

5.55 pd_packet_extd_t Struct Reference

```
#include <pd.h>
```

Data Fields

- [uint8_t sop](#)
- [uint8_t len](#)
- [uint8_t msg](#)
- [uint8_t data_role](#)
- [pd_hdr_t](#) [hdr](#)
- [pd_do_t](#) [dat](#) [[MAX_EXTD_PKT_WORDS](#)]

5.55.1 Detailed Description

Struct to hold extended PD packets (messages).

5.55.2 Field Documentation

5.55.2.1 dat

```
pd_do_t dat [MAX_EXTD_PKT_WORDS]
```

Data associated with the message.

5.55.2.2 data_role

```
uint8_t data_role
```

Data role.

5.55.2.3 hdr

```
pd_hdr_t hdr
```

Message header, including extended header.

5.55.2.4 len

```
uint8_t len
```

Length of the message: Unused for unchunked extended messages.

5.55.2.5 msg

```
uint8_t msg
```

Message code.

5.55.2.6 sop

```
uint8_t sop
```

Packet type.

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

- [pd_common/pd.h](#)

5.56 pd_packet_t Struct Reference

```
#include <pd.h>
```

Data Fields

- [uint8_t sop](#)
- [uint8_t len](#)
- [uint8_t msg](#)
- [uint8_t data_role](#)
- [pd_hdr_t hdr](#)
- [pd_do_t dat](#) [[MAX_NO_OF_DO](#)]

5.56.1 Detailed Description

Struct to hold a PD packet.

5.56.2 Field Documentation

5.56.2.1 dat

[pd_do_t](#) [dat](#) [[MAX_NO_OF_DO](#)]

Data objects associated with the message.

5.56.2.2 data_role

[uint8_t](#) [data_role](#)

Data role.

5.56.2.3 hdr

[pd_hdr_t](#) [hdr](#)

Message header.

5.56.2.4 len

[uint8_t](#) [len](#)

Length in data objects.

5.56.2.5 msg

[uint8_t](#) [msg](#)

Message code.

5.56.2.6 sop

[uint8_t](#) [sop](#)

Packet type.

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

- [pd_common/pd.h](#)

5.57 pd_port_config_t Struct Reference

```
#include <pd.h>
```

Data Fields

- uint16_t id_vdm_offset
- uint16_t id_vdm_length
- uint16_t svid_vdm_offset
- uint16_t svid_vdm_length
- uint16_t mode_vdm_offset
- uint16_t mode_vdm_length
- uint16_t ext_src_cap_offset
- uint16_t ext_src_cap_length
- uint16_t ext_snk_cap_offset
- uint16_t ext_snk_cap_length
- uint16_t reserved_0 [2]
- uint16_t ra_timeout
- uint16_t reserved_1 [3]
- uint8_t port_role
- uint8_t default_role
- uint8_t current_level
- uint8_t is_src_bat
- uint8_t is_snk_bat
- uint8_t snk_usb_susp_en
- uint8_t snk_usb_comm_en
- uint8_t volatile swap_response
- uint8_t drp_toggle_en
- uint8_t src_pdo_cnt
- uint8_t default_src_pdo_mask
- uint8_t snk_pdo_cnt
- uint8_t default_sink_pdo_mask
- uint8_t rp_supported
- uint8_t pd_operation_en
- uint8_t try_src_en
- uint8_t cable_disc_en
- uint8_t dead_bat_support
- uint8_t err_recovery_en
- uint8_t volatile port_disable
- uint8_t volatile frs_enable
- uint8_t volatile vconn_retain
- uint16_t reserved_3 [5]
- uint32_t src_pdo_list [MAX_NO_OF_PDO]
- uint32_t snk_pdo_list [MAX_NO_OF_PDO]
- uint16_t snk_pdo_max_min_current_pwr [MAX_NO_OF_PDO]
- uint16_t reserved_4
- uint8_t volatile protection_enable
- uint8_t reserved_8
- uint16_t ovp_tbl_offset
- uint16_t ocp_tbl_offset
- uint16_t uvp_tbl_offset
- uint16_t scp_tbl_offset
- uint16_t vconn_ocp_tbl_offset
- uint16_t otp_tbl_offset

- uint16_t pwr_tbl_offset
- uint16_t chg_cfg_tbl_offset
- uint16_t bat_chg_tbl_offset
- uint16_t rcp_tbl_offset
- uint8_t reserved_9 [2]
- uint8_t volatile dp_config_supported
- uint8_t volatile dp_mux_control
- uint8_t volatile alt_mode_trigger
- uint8_t dp_oper
- uint8_t volatile dp_pref_mode
- uint8_t reserved_6 [3]
- uint16_t bb_tbl_offset
- uint8_t volatile type_a_enable
- uint8_t reserved_10
- uint16_t type_a_pwr_tbl_offset
- uint16_t type_a_chg_tbl_offset
- uint16_t dock_cfg_tbl_offset
- uint16_t spm_cfg_tbl_offset
- uint16_t auto_cfg_tbl_offset
- uint16_t tbthost_cfg_tbl_offset
- uint16_t alt_mode_tbl_offset
- uint16_t custom_host_tbl_offset
- uint16_t icl_tgl_tbl_offset
- uint16_t custom_alt_mode_tbl_offset
- uint8_t reserved_11 [48]

5.57.1 Detailed Description

PD port-specific configuration data from the configuration table.

5.57.2 Field Documentation

5.57.2.1 alt_mode_tbl_offset

```
uint16_t alt_mode_tbl_offset
```

Byte 0xB8: Offset to alternate modes configuration table.

5.57.2.2 alt_mode_trigger

```
uint8_t volatile alt_mode_trigger
```

Byte 0xA2: ALT_MODE_TRIGGER: Trigger to enable/disable alt modes. Bit position of trigger mask corresponds to alt modes index in compatibility_mode_table from alt_modes_config.h file. Bit value 0 => Alternate mode will be entered automatically. Bit value 1 => Alternate mode manager will be waiting for HPI Enter mode command to enter alt mode.

5.57.2.3 auto_cfg_tbl_offset

```
uint16_t auto_cfg_tbl_offset
```

Byte 0xB4: Offset to automotive charger settings table.

5.57.2.4 bat_chg_tbl_offset

uint16_t bat_chg_tbl_offset

Byte 0x9A: Offset to battery charging parameters.

5.57.2.5 bb_tbl_offset

uint16_t bb_tbl_offset

Byte 0xA8: Two byte offset to Billboard settings

5.57.2.6 cable_disc_en

uint8_t cable_disc_en

Byte 0x30: Whether cable discovery is supported on the port.

5.57.2.7 chg_cfg_tbl_offset

uint16_t chg_cfg_tbl_offset

Byte 0x98: Offset to legacy charging parameters.

5.57.2.8 current_level

uint8_t current_level

Byte 0x22: Type-C current level: 0=Default, 1=1.5A, 2=3A.

5.57.2.9 custom_alt_mode_tbl_offset

uint16_t custom_alt_mode_tbl_offset

Byte 0xBE: Offset to custom alternate mode configuration table.

5.57.2.10 custom_host_tbl_offset

uint16_t custom_host_tbl_offset

Byte 0xBA: Offset to custom Host configuration table.

5.57.2.11 dead_bat_support

uint8_t dead_bat_support

Byte 0x31: Whether firmware should force Sink operation in Dead Battery condition.

5.57.2.12 default_role

uint8_t default_role

Byte 0x21: Default port role in case of a dual role port: 0=Sink, 1=Source.

5.57.2.13 default_sink_pdo_mask

uint8_t default_sink_pdo_mask

Byte 0x2C: Default Sink PDO enable mask.

5.57.2.14 default_src_pdo_mask

uint8_t default_src_pdo_mask

Byte 0x2A: Default Source PDO enable mask.

5.57.2.15 dock_cfg_tbl_offset

uint16_t dock_cfg_tbl_offset

Byte 0xB0: Offset to dock configuration parameter table.

5.57.2.16 dp_config_supported

uint8_t volatile dp_config_supported

Byte 0xA0: Supported Pin configurations for DP modes 0b00000000: USB SS only. 0b00000001: Reserved for future use (A). 0b00000010: Reserved for future use (B). 0b00000100: Pin Config C. 0b00001000: Pin Config D. 0b00010000: Pin Config E. 0b00100000: Pin Config F.

5.57.2.17 dp_mux_control

uint8_t volatile dp_mux_control

Byte 0xA1: DP_MUX_CONTROL method: 0 => DP MUX Controlled by CCG. 1 => DP MUX Controlled by EC.

5.57.2.18 dp_oper

uint8_t dp_oper

Byte 0xA3: Type of DP operation supported. Bit 0: DP Sink supported Bit 1: DP Source supported.

5.57.2.19 dp_pref_mode

uint8_t volatile dp_pref_mode

Byte 0xA4: DP preferred mode. Bit 0: 0: CCG as DP Sink prefers 4 lane DP mode only. 1: CCG as DP Sink prefers 2 lane DP + USB SS Mode. All other bits are reserved.

5.57.2.20 drp_toggle_en

uint8_t drp_toggle_en

Byte 0x28: Whether DRP toggle is enabled.

5.57.2.21 err_recovery_en

uint8_t err_recovery_en

Byte 0x32: Whether PD error recovery is enabled.

5.57.2.22 ext_snk_cap_length

```
uint16_t ext_snk_cap_length
```

Byte 0x12: Two byte length of the Snk. Cap Extended response.

5.57.2.23 ext_snk_cap_offset

```
uint16_t ext_snk_cap_offset
```

Byte 0x10: Two byte offset to the Snk. Cap Extended response.

5.57.2.24 ext_src_cap_length

```
uint16_t ext_src_cap_length
```

Byte 0x0E: Two byte length of the Src. Cap Extended response.

5.57.2.25 ext_src_cap_offset

```
uint16_t ext_src_cap_offset
```

Byte 0x0C: Two byte offset to the Src. Cap Extended response.

5.57.2.26 frs_enable

```
uint8_t volatile frs_enable
```

Byte 0x34: Fast Role Swap enable flags.

5.57.2.27 icl_tgl_tbl_offset

```
uint16_t icl_tgl_tbl_offset
```

Byte 0xBC: Offset to ICL/TGL configuration table.

5.57.2.28 id_vdm_length

```
uint16_t id_vdm_length
```

Byte 0x02: Two byte length of the Discover ID Response VDM.

5.57.2.29 id_vdm_offset

```
uint16_t id_vdm_offset
```

Byte 0x00: Two byte offset to the Discover ID Response VDM.

5.57.2.30 is_snk_bat

```
uint8_t is_snk_bat
```

Byte 0x24: Whether the power sink is connected to a battery.

5.57.2.31 is_src_bat

```
uint8_t is_src_bat
```

Byte 0x23: Whether the power source is connected to a battery.

5.57.2.32 mode_vdm_length

```
uint16_t mode_vdm_length
```

Byte 0x0A: Two byte length of the Discover Mode Response VDM.

5.57.2.33 mode_vdm_offset

```
uint16_t mode_vdm_offset
```

Byte 0x08: Two byte offset to the Discover Mode Response VDM.

5.57.2.34 ocp_tbl_offset

```
uint16_t ocp_tbl_offset
```

Byte 0x8C: Offset to VBus OCP settings.

5.57.2.35 otp_tbl_offset

```
uint16_t otp_tbl_offset
```

Byte 0x94: Offset to OTP settings.

5.57.2.36 ovp_tbl_offset

```
uint16_t ovp_tbl_offset
```

Byte 0x8A: Offset to VBus OVP settings.

5.57.2.37 pd_operation_en

```
uint8_t pd_operation_en
```

Byte 0x2E: Whether PD operation is supported on the port.

5.57.2.38 port_disable

```
uint8_t volatile port_disable
```

Byte 0x33: Port disable flag.

5.57.2.39 port_role

```
uint8_t port_role
```

Byte 0x20: PD port role: 0=Sink, 1=Source, 2=Dual Role.

5.57.2.40 protection_enable

uint8_t volatile protection_enable

Byte 0x88: Enable field for protection settings Bit0: ovp enable Bit1: ocp enable Bit2: uvp enable Bit3: scp enable Bit4: vconn ocp enable Bit5: otp enable Bit(6:7): Reserved for future use

5.57.2.41 pwr_tbl_offset

uint16_t pwr_tbl_offset

Byte 0x96: Offset to power parameters.

5.57.2.42 ra_timeout

uint16_t ra_timeout

Byte 0x18: Ra removal delay for EMCA applications, measured in ms.

5.57.2.43 rcp_tbl_offset

uint16_t rcp_tbl_offset

Byte 0x9C: Offset to VBus RCP settings.

5.57.2.44 reserved_0

uint16_t reserved_0[2]

Byte 0x14: Reserved area for additional VDMs.

5.57.2.45 reserved_1

uint16_t reserved_1[3]

Byte 0x1A: Reserved area for EMCA configuration.

5.57.2.46 reserved_10

uint8_t reserved_10

Byte 0xAB: Reserved.

5.57.2.47 reserved_11

uint8_t reserved_11[48]

Byte 0xC0: Reserved area for future expansion.

5.57.2.48 reserved_3

uint16_t reserved_3[5]

Byte 0x36: Reserved words for padding to 4-byte aligned address.

5.57.2.49 reserved_4

```
uint16_t reserved_4
```

Byte 0x86: Reserved space for additional port parameters.

5.57.2.50 reserved_6

```
uint8_t reserved_6[3]
```

Byte 0xA5: Reserved area for future expansion.

5.57.2.51 reserved_8

```
uint8_t reserved_8
```

Byte 0x89: Reserved for future use.

5.57.2.52 reserved_9

```
uint8_t reserved_9[2]
```

Byte 0x9E: Reserved.

5.57.2.53 rp_supported

```
uint8_t rp_supported
```

Byte 0x2D: Supported Rp values. Multiple bits can be set. Bit 0 => Default current support. Bit 1 => 1.5A support. Bit 2 => 3A support.

5.57.2.54 scp_tbl_offset

```
uint16_t scp_tbl_offset
```

Byte 0x90: Offset to VBus SCP settings.

5.57.2.55 snk_pdo_cnt

```
uint8_t snk_pdo_cnt
```

Byte 0x2B: Number of valid sink PDOs in the table: Maximum supported value is 7.

5.57.2.56 snk_pdo_list

```
uint32_t snk_pdo_list[MAX_NO_OF_PDO]
```

Byte 0x5C: Sink PDO data array.

5.57.2.57 snk_pdo_max_min_current_pwr

```
uint16_t snk_pdo_max_min_current_pwr[MAX_NO_OF_PDO]
```

Byte 0x78: Array of sink PDO parameters. For each element, the format is as below: Bit 15 => Give Back support flag Bits 14:0 => Minimum sink operating current.

5.57.2.58 snk_usb_comm_en

```
uint8_t snk_usb_comm_en
```

Byte 0x26: Whether USB communication is supported.

5.57.2.59 snk_usb_susp_en

```
uint8_t snk_usb_susp_en
```

Byte 0x25: Whether USB suspend is supported.

5.57.2.60 spm_cfg_tbl_offset

```
uint16_t spm_cfg_tbl_offset
```

Byte 0xB2: Offset to Source Policy Manager parameter table.

5.57.2.61 src_pdo_cnt

```
uint8_t src_pdo_cnt
```

Byte 0x29: Number of valid source PDOs in the table: Maximum supported value is 7.

5.57.2.62 src_pdo_list

```
uint32_t src_pdo_list[MAX_NO_OF_PDO]
```

Byte 0x40: Source PDO data array.

5.57.2.63 svid_vdm_length

```
uint16_t svid_vdm_length
```

Byte 0x06: Two byte length of the Discover SVID Response VDM.

5.57.2.64 svid_vdm_offset

```
uint16_t svid_vdm_offset
```

Byte 0x04: Two byte offset to the Discover SVID Response VDM.

5.57.2.65 swap_response

```
uint8_t volatile swap_response
```

Byte 0x27: Response to be sent for each USB-PD SWAP command: Bits 1:0 => DR_SWAP response Bits 3:2 => PR_SWAP response Bits 5:4 => VCONN_SWAP response Allowed values are: 0=ACCEPT, 1=REJECT, 2=WAIT.

5.57.2.66 tbthost_cfg_tbl_offset

```
uint16_t tbthost_cfg_tbl_offset
```

Byte 0xB6: Offset to Thunderbolt Host config parameter table.

5.57.2.67 try_src_en

```
uint8_t try_src_en
```

Byte 0x2F: Whether Try.SRC state is supported on the port.

5.57.2.68 type_a_chg_tbl_offset

```
uint16_t type_a_chg_tbl_offset
```

Byte 0xAE: Offset to battery charging parameters of Type-A port.

5.57.2.69 type_a_enable

```
uint8_t volatile type_a_enable
```

Byte 0xAA: Type-A port support.

5.57.2.70 type_a_pwr_tbl_offset

```
uint16_t type_a_pwr_tbl_offset
```

Byte 0xAC: Offset to power parameters of Type-A port.

5.57.2.71 uvp_tbl_offset

```
uint16_t uvp_tbl_offset
```

Byte 0x8E: Offset to VBus UVP settings.

5.57.2.72 vconn_ocp_tbl_offset

```
uint16_t vconn_ocp_tbl_offset
```

Byte 0x92: Offset to Vcon OCP settings.

5.57.2.73 vconn_retain

```
uint8_t volatile vconn_retain
```

Byte 0x35: Whether VConn should be retained in ON state.

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

- [pd_common/pd.h](#)

5.58 pd_port_status_t::PD_PORT_STAT Struct Reference

```
#include <pd.h>
```

Data Fields

- [uint32_t dflt_data_role](#): 2
- [uint32_t dflt_data_pref](#): 1
- [uint32_t dflt_power_role](#): 2

- uint32_t `dflt_power_pref`: 1
- uint32_t `cur_data_role`: 1
- uint32_t `reserved0`: 1
- uint32_t `cur_power_role`: 1
- uint32_t `min_state`: 1
- uint32_t `contract_exist`: 1
- uint32_t `emca_present`: 1
- uint32_t `vconn_src`: 1
- uint32_t `vconn_on`: 1
- uint32_t `rp_status`: 1
- uint32_t `pe_rdy`: 1
- uint32_t `ccg_spec_rev`: 2
- uint32_t `peer_pd3_supp`: 1
- uint32_t `peer_unchunk_supp`: 1
- uint32_t `emca_spec_rev`: 2
- uint32_t `emca_type`: 1
- uint32_t `reserved2`: 9

5.58.1 Detailed Description

Structure containing status bits.

5.58.2 Field Documentation

5.58.2.1 `ccg_spec_rev`

`uint32_t ccg_spec_rev`

USB-PD revision supported by CCG firmware.

5.58.2.2 `contract_exist`

`uint32_t contract_exist`

Whether explicit contract exists.

5.58.2.3 `cur_data_role`

`uint32_t cur_data_role`

Current data role.

5.58.2.4 `cur_power_role`

`uint32_t cur_power_role`

Current power role.

5.58.2.5 `dflt_data_pref`

`uint32_t dflt_data_pref`

Preferred data role in case of DRP.

5.58.2.6 dflt_data_role

uint32_t dflt_data_role

Default data role.

5.58.2.7 dflt_power_pref

uint32_t dflt_power_pref

Preferred power role in case of DRP.

5.58.2.8 dflt_power_role

uint32_t dflt_power_role

Default power role.

5.58.2.9 emca_present

uint32_t emca_present

EMCA detected or not.

5.58.2.10 emca_spec_rev

uint32_t emca_spec_rev

USB-PD revision supported by cable marker.

5.58.2.11 emca_type

uint32_t emca_type

EMCA type: Passive=0, Active=1.

5.58.2.12 min_state

uint32_t min_state

Whether in Min state (due to GotoMin).

5.58.2.13 pe_rdy

uint32_t pe_rdy

Whether Policy Engine is in Ready state.

5.58.2.14 peer_pd3_supp

uint32_t peer_pd3_supp

Whether port partner supports PD 3.0.

5.58.2.15 peer_unchunk_supp

uint32_t peer_unchunk_supp

Whether port partner supports unchunked messages.

5.58.2.16 reserved0

uint32_t reserved0

Reserved.

5.58.2.17 reserved2

uint32_t reserved2

Reserved field.

5.58.2.18 rp_status

uint32_t rp_status

Current Rp status.

5.58.2.19 vconn_on

uint32_t vconn_on

Whether VConn is actually ON.

5.58.2.20 vconn_src

uint32_t vconn_src

Whether CCG is VConn source.

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

- [pd_common/pd.h](#)

5.59 pd_port_status_t Union Reference

```
#include <pd.h>
```

Data Structures

- struct [PD_PORT_STAT](#)

Data Fields

- uint32_t [val](#)
- struct [pd_port_status_t::PD_PORT_STAT](#) [status](#)

5.59.1 Detailed Description

PD port status as reported to Embedded Controller.

5.59.2 Field Documentation

5.59.2.1 `status`

```
struct pd_port_status_t::PD_PORT_STAT status
```

PD port status structure.

5.59.2.2 `val`

```
uint32_t val
```

PD-Status register value.

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

- [pd_common/pd.h](#)

5.60 `pd_power_status_t` Struct Reference

```
#include <pd.h>
```

Data Fields

- [uint8_t `intl_temperature`](#)
- [uint8_t `present_input`](#)
- [uint8_t `battery_input`](#)
- [uint8_t `event_flags`](#)
- [uint8_t `temp_status`](#)
- [uint8_t `power_status`](#)
- [uint8_t `dummy` \[2\]](#)

5.60.1 Detailed Description

PD port status corresponding to the Status Data Block (SSDB) See Table 6-39 of USB-PD R3 specification.

5.60.2 Field Documentation

5.60.2.1 `battery_input`

```
uint8_t battery_input
```

Reports the current battery status.

5.60.2.2 dummy

```
uint8_t dummy[2]
```

Reserved field used for 4 byte alignment.

5.60.2.3 event_flags

```
uint8_t event_flags
```

Event flags.

5.60.2.4 intl_temperature

```
uint8_t intl_temperature
```

Port's internal temperature. 0 if not supported.

5.60.2.5 power_status

```
uint8_t power_status
```

Power status.

5.60.2.6 present_input

```
uint8_t present_input
```

Reports current input power status.

5.60.2.7 temp_status

```
uint8_t temp_status
```

Temperature status.

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

- [pd_common/pd.h](#)

5.61 pd_stack_conf_t Struct Reference

```
#include <pd.h>
```

Data Fields

- bool [source_only](#)
- bool [pd_rev_3](#)
- bool [frs_rx](#)
- bool [frs_tx](#)

5.61.1 Detailed Description

Struct used to fetch the PD stack configuration supported by the library that is currently in use.

5.61.2 Field Documentation

5.61.2.1 frs_rx

bool frs_rx

- True : FRS receive is enabled
- False : FRS receive is disabled

5.61.2.2 frs_tx

bool frs_tx

- True : FRS transmit is enabled
- False : FRS transmit is disabled

5.61.2.3 pd_rev_3

bool pd_rev_3

- True : CCG_PD_REV3_ENABLE macro is enabled
- False : CCG_PD_REV3_ENABLE macro is disabled

5.61.2.4 source_only

bool source_only

- True : CCG_SOURCE_ONLY macro is enabled
- False : CCG_SOURCE_ONLY macro is disabled

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

- [pd_common/pd.h](#)

5.62 pd_status_t Struct Reference

```
#include <pd_protocol.h>
```

Data Fields

- [pd_cbk_t](#) cbk
- [prl_cntrs_t](#) ctrs [MAX_SOP_TYPES]
- [uint32_t](#) tx_buf [MAX_PD_PKT_WORDS]
- [uint32_t](#) tx_header
- [pd_packet_extd_t](#) rx_packet
- [uint8_t](#) cur_rec_msg_id
- [sop_t](#) last_rcvd_sop
- [uint8_t](#) volatile avoid_retry
- [uint8_t](#) volatile tx_busy
- [sop_t](#) tx_sop
- [pd_extd_hdr_t](#) tx_extd_hdr
- [bool](#) tx_extd
- [sop_t](#) last_tx_sop
- [uint8_t](#) tx_msg_type
- [uint8_t](#) tx_count
- [uint8_t](#) bist_test_en
- [uint32_t](#) volatile rx_evt
- [bool](#) volatile rx_busy
- [volatile uint8_t](#) rev3_enable
- [volatile uint8_t](#) frs_tx_enable
- [volatile uint8_t](#) frs_rx_enable

5.62.1 Detailed Description

Structure to hold protocol layer status.

Warning: Changes to this structure will break compatibility with the ROM-ed version of the Protocol layer on the CCG6 device, and can cause application malfunction. This restriction only applies to projects that use the ROM-ed protocol layer code on the CCG6 device family.

5.62.2 Field Documentation

5.62.2.1 avoid_retry

```
uint8_t volatile avoid_retry
```

Flag to skip retry on CRCReceiveTimer expiry.

5.62.2.2 bist_test_en

```
uint8_t bist_test_en
```

Flag indicating BIST test data is enabled.

5.62.2.3 cbk

```
pd_cbk_t cbk
```

Callback used to send notifications to Policy Engine.

5.62.2.4 ctrs

```
prl_cntrs_t ctrs[MAX_SOP_TYPES]
```

Protocol counters for various packet types.

5.62.2.5 cur_rec_msg_id

```
uint8_t cur_rec_msg_id
```

Stores message ID (excluding GoodCRC) of the received message.

5.62.2.6 frs_rx_enable

```
volatile uint8_t frs_rx_enable
```

Flag indicating that FRS receive support is enabled.

5.62.2.7 frs_tx_enable

```
volatile uint8_t frs_tx_enable
```

Flag indicating that FRS transmit support is enabled.

5.62.2.8 last_rcvd_sop

```
sop_t last_rcvd_sop
```

Stores last received SOP type.

5.62.2.9 last_tx_sop

```
sop_t last_tx_sop
```

Stores the SOP type of last transmitted message.

5.62.2.10 rev3_enable

```
volatile uint8_t rev3_enable
```

Flag indicating the PD Revision 3.0 support is enabled.

5.62.2.11 rx_busy

```
bool volatile rx_busy
```

Flag indicating RX state machine is busy.

5.62.2.12 rx_evt

```
uint32_t volatile rx_evt
```

Type of receive state machine event received from the HAL.

5.62.2.13 rx_packet

`pd_packet_extd_t rx_packet`

Buffer used to hold received PD message. Can be extended or not.

5.62.2.14 tx_buf

`uint32_t tx_buf[MAX_PD_PKT_WORDS]`

Buffer used to hold message to be (or being) transmitted.

5.62.2.15 tx_busy

`uint8_t volatile tx_busy`

Flag indicating TX state machine is busy.

5.62.2.16 tx_count

`uint8_t tx_count`

Holds the data object count of the message being transmitted.

5.62.2.17 tx_extd

`bool tx_extd`

Flag that indicates that the current message is extended.

5.62.2.18 tx_extd_hdr

`pd_extd_hdr_t tx_extd_hdr`

Holds the extended header for the message being transmitted.

5.62.2.19 tx_header

`uint32_t tx_header`

PD header corresponding to the message being transmitted.

5.62.2.20 tx_msg_type

`uint8_t tx_msg_type`

Stores the message type of message being transmitted.

5.62.2.21 tx_sop

`sop_t tx_sop`

Stores the SOP type of message being transmitted.

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

- [pd_common/pd_protocol.h](#)

5.63 pd_do_t::PPS_SNK Struct Reference

```
#include <pd.h>
```

Data Fields

- uint32_t `op_cur`: 7
- uint32_t `rsvd1`: 1
- uint32_t `min_volt`: 8
- uint32_t `rsvd2`: 1
- uint32_t `max_volt`: 8
- uint32_t `rsvd3`: 1
- uint32_t `cur_fb`: 1
- uint32_t `rsvd4`: 1
- uint32_t `apdo_type`: 2
- uint32_t `supply_type`: 2

5.63.1 Detailed Description

Programmable Power Supply Sink PDO.

5.63.2 Field Documentation

5.63.2.1 apdo_type

```
uint32_t apdo_type
```

APDO type: Should be 0 for PPS.

5.63.2.2 cur_fb

```
uint32_t cur_fb
```

Whether current foldback is required.

5.63.2.3 max_volt

```
uint32_t max_volt
```

Maximum voltage in 100 mV units.

5.63.2.4 min_volt

```
uint32_t min_volt
```

Minimum voltage in 100 mV units.

5.63.2.5 op_cur

```
uint32_t op_cur
```

Operating current in 50 mA units.

5.63.2.6 rsvd1

uint32_t rsvd1

Reserved field.

5.63.2.7 rsvd2

uint32_t rsvd2

Reserved field.

5.63.2.8 rsvd3

uint32_t rsvd3

Reserved field.

5.63.2.9 rsvd4

uint32_t rsvd4

Reserved field.

5.63.2.10 supply_type

uint32_t supply_type

PDO type: Should be 3 for PPS APDO.

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

- [pd_common/pd.h](#)

5.64 pd_do_t::PPS_SRC Struct Reference

```
#include <pd.h>
```

Data Fields

- uint32_t [max_cur](#): 7
- uint32_t [rsvd1](#): 1
- uint32_t [min_volt](#): 8
- uint32_t [rsvd2](#): 1
- uint32_t [max_volt](#): 8
- uint32_t [rsvd3](#): 2
- uint32_t [pps_pwr_limited](#): 1
- uint32_t [apdo_type](#): 2
- uint32_t [supply_type](#): 2

5.64.1 Detailed Description

Programmable Power Supply Source PDO.

5.64.2 Field Documentation

5.64.2.1 apdo_type

uint32_t apdo_type

APDO type: Should be 0 for PPS.

5.64.2.2 max_cur

uint32_t max_cur

Maximum current in 50 mA units.

5.64.2.3 max_volt

uint32_t max_volt

Maximum voltage in 100 mV units.

5.64.2.4 min_volt

uint32_t min_volt

Minimum voltage in 100 mV units.

5.64.2.5 pps_pwr_limited

uint32_t pps_pwr_limited

Whether PPS power has been limited.

5.64.2.6 rsvd1

uint32_t rsvd1

Reserved field.

5.64.2.7 rsvd2

uint32_t rsvd2

Reserved field.

5.64.2.8 rsvd3

uint32_t rsvd3

Reserved field.

5.64.2.9 supply_type

```
uint32_t supply_type
```

PDO type: Should be 3 for PPS APDO.

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

- [pd_common/pd.h](#)

5.65 prl_cntrs_t Struct Reference

```
#include <pd_protocol.h>
```

Data Fields

- [uint8_t tr_msg_id](#)
- [uint8_t rec_msg_id](#)
- [uint8_t volatile first_msg_rcvd](#)

5.65.1 Detailed Description

Struct to hold PD protocol message IDs and flags. Separate structures need to be maintained for each packet type (SOP, SOP' and SOP").

5.65.2 Field Documentation

5.65.2.1 first_msg_rcvd

```
uint8_t volatile first_msg_rcvd
```

Flag that indicates whether any message has been received so far.

5.65.2.2 rec_msg_id

```
uint8_t rec_msg_id
```

The message ID of last received message.

5.65.2.3 tr_msg_id

```
uint8_t tr_msg_id
```

The message ID to be used on the next transmitted message.

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

- [pd_common/pd_protocol.h](#)

5.66 pwr_params_t Struct Reference

```
#include <pd.h>
```

Data Fields

- uint8_t table_len
- uint8_t fb_type
- uint16_t vbus_min_volt
- uint16_t vbus_max_volt
- uint16_t vbus_dflt_volt
- uint32_t fb_ctrl_r1
- uint32_t fb_ctrl_r2
- uint16_t cable_resistance
- uint16_t vbus_offset_volt
- uint8_t cur_sense_res
- uint8_t src_gate_drv_str
- uint8_t reserved_0 [2]
- uint16_t vbtr_up_step_width
- uint16_t vbtr_down_step_width
- uint8_t prim_sec_turns_ratio
- uint8_t sr_enable
- uint8_t sr_rise_time
- uint8_t sr_fall_time
- uint8_t sr_async_thresh
- uint8_t sr_supply_doubler
- uint8_t reserved_2 [2]
- uint8_t pwm_mode
- uint8_t pwm_min_freq
- uint8_t pwm_max_freq
- uint8_t pwm_fix_freq
- uint8_t max_pwm_duty_cycle
- uint8_t reserved_3 [3]

5.66.1 Detailed Description

Struct to hold the port power parameters.

5.66.2 Field Documentation

5.66.2.1 cable_resistance

```
uint16_t cable_resistance
```

Cable resistance in mOhm

5.66.2.2 cur_sense_res

```
uint8_t cur_sense_res
```

Available to adjust CSA accuracy on board. Unit of 0.1mOhm

5.66.2.3 fb_ctrl_r1

```
uint32_t fb_ctrl_r1
```

[DEPRECATED]: Feedback control circuit: R1 resistance in Ohms. Not to be used.

5.66.2.4 fb_ctrl_r2

uint32_t fb_ctrl_r2

[DEPRECATED]: Feedback control circuit: R2 resistance in Ohms. Not to be used.

5.66.2.5 fb_type

uint8_t fb_type

Type of power feedback: Bit 0 -> No feedback Bit 1 -> PWM Bit 2 -> Direct feedback Bit 3 -> Opto-isolator based feedback

5.66.2.6 max_pwm_duty_cycle

uint8_t max_pwm_duty_cycle

Maximum allowed PWM pulse duty cycle

5.66.2.7 prim_sec_turns_ratio

uint8_t prim_sec_turns_ratio

Primary to secondary turns ratio rounded to nearest decimal

5.66.2.8 pwm_fix_freq

uint8_t pwm_fix_freq

PWM switching frequency in FF mode in KHz

5.66.2.9 pwm_max_freq

uint8_t pwm_max_freq

Maximum allowed switching frequency in QR/QR+FF mode in KHz

5.66.2.10 pwm_min_freq

uint8_t pwm_min_freq

Minimum allowed switching frequency in QR/QR+FF mode in KHz

5.66.2.11 pwm_mode

uint8_t pwm_mode

Indicates operational mode of power adapter secondary controller

5.66.2.12 reserved_0

uint8_t reserved_0[2]

Reserved for future use

5.66.2.13 reserved_2`uint8_t reserved_2[2]`

Reserved for future use

5.66.2.14 reserved_3`uint8_t reserved_3[3]`

Reserved for future use

5.66.2.15 sr_async_thresh`uint8_t sr_async_thresh`

Parameter in 0.1us

5.66.2.16 sr_enable`uint8_t sr_enable`

Enable/Disable the SR controller

5.66.2.17 sr_fall_time`uint8_t sr_fall_time`

SR gate driver fall time configuration

5.66.2.18 sr_rise_time`uint8_t sr_rise_time`

SR gate driver rise time configuration

5.66.2.19 sr_supply_doubler`uint8_t sr_supply_doubler`

Enable/Disable the doubler for gate drive function

5.66.2.20 src_gate_drv_str`uint8_t src_gate_drv_str`

Vbus source gate drive strength

5.66.2.21 table_len`uint8_t table_len`

Power parameters table length in bytes

5.66.2.22 vbtr_down_step_width

```
uint16_t vbtr_down_step_width
```

Vbtr Downward transition step width in 1us units

5.66.2.23 vbtr_up_step_width

```
uint16_t vbtr_up_step_width
```

Vbtr Upward transition step width in 1us units

5.66.2.24 vbus_dflt_volt

```
uint16_t vbus_dflt_volt
```

Default VBus supply voltage when feedback control is tri-stated.

5.66.2.25 vbus_max_volt

```
uint16_t vbus_max_volt
```

VBus maximum voltage in mV

5.66.2.26 vbus_min_volt

```
uint16_t vbus_min_volt
```

VBus minimum voltage in mV

5.66.2.27 vbus_offset_volt

```
uint16_t vbus_offset_volt
```

VBus offset voltage in addition to contracted voltage in mV

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

- [pd_common/pd.h](#)

5.67 pd_do_t::QC_4_0_DATA_VDO Struct Reference

```
#include <pd.h>
```

Data Fields

- `uint32_t data_0`: 8
- `uint32_t data_1`: 8
- `uint32_t data_2`: 8
- `uint32_t data_3`: 8

5.67.1 Detailed Description

Structure representing an Unstructured VDM data object as defined by QC 4.0 spec.

5.67.2 Field Documentation

5.67.2.1 data_0

uint32_t data_0

Command data #0.

5.67.2.2 data_1

uint32_t data_1

Command data #1.

5.67.2.3 data_2

uint32_t data_2

Command data #2.

5.67.2.4 data_3

uint32_t data_3

Command data #3.

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

- [pd_common/pd.h](#)

5.68 rcp_settings_t Struct Reference

```
#include <pd.h>
```

Data Fields

- uint8_t [table_len](#)
- uint8_t [retry_cnt](#)
- uint16_t [resvd](#)

5.68.1 Detailed Description

Struct to hold the RCP settings.

5.68.2 Field Documentation

5.68.2.1 resvd

```
uint16_t resvd
```

Reserved bytes.

5.68.2.2 retry_cnt

```
uint8_t retry_cnt
```

Number of consecutive ORP events allowed before the port is suspended by CCG firmware.

5.68.2.3 table_len

```
uint8_t table_len
```

Table length in bytes

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

- [pd_common/pd.h](#)

5.69 pd_do_t::RDO_BAT Struct Reference

```
#include <pd.h>
```

Data Fields

- uint32_t [max_op_power](#): 10
- uint32_t [op_power](#): 10
- uint32_t [rsrvd1](#): 3
- uint32_t [unchunk_sup](#): 1
- uint32_t [no_usb_suspend](#): 1
- uint32_t [usb_comm_cap](#): 1
- uint32_t [cap_mismatch](#): 1
- uint32_t [give_back_flag](#): 1
- uint32_t [obj_pos](#): 3
- uint32_t [rsrvd2](#): 1

5.69.1 Detailed Description

Structure representing a Battery Request Data Object.

5.69.2 Field Documentation

5.69.2.1 cap_mismatch

```
uint32_t cap_mismatch
```

Capability mismatch.

5.69.2.2 give_back_flag

uint32_t give_back_flag

GiveBack flag = 0.

5.69.2.3 max_op_power

uint32_t max_op_power

Maximum operating power in 250mW units.

5.69.2.4 no_usb_suspend

uint32_t no_usb_suspend

No USB suspend.

5.69.2.5 obj_pos

uint32_t obj_pos

Object position.

5.69.2.6 op_power

uint32_t op_power

Operating power in 250mW units.

5.69.2.7 rsrvd1

uint32_t rsrvd1

Reserved field.

5.69.2.8 rsrvd2

uint32_t rsrvd2

Reserved field.

5.69.2.9 unchunk_sup

uint32_t unchunk_sup

Unchunked extended messages supported.

5.69.2.10 usb_comm_cap

uint32_t usb_comm_cap

USB communication capability.

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

- [pd_common/pd.h](#)

5.70 pd_do_t::RDO_BAT_GIVEBACK Struct Reference

```
#include <pd.h>
```

Data Fields

- uint32_t `min_op_power`: 10
- uint32_t `op_power`: 10
- uint32_t `rsrvd1`: 3
- uint32_t `unchunk_sup`: 1
- uint32_t `no_usb_suspend`: 1
- uint32_t `usb_comm_cap`: 1
- uint32_t `cap_mismatch`: 1
- uint32_t `give_back_flag`: 1
- uint32_t `obj_pos`: 3
- uint32_t `rsrvd2`: 1

5.70.1 Detailed Description

Structure representing a Battery Request Data Object with GiveBack.

5.70.2 Field Documentation

5.70.2.1 cap_mismatch

```
uint32_t cap_mismatch
```

Capability mismatch.

5.70.2.2 give_back_flag

```
uint32_t give_back_flag
```

GiveBack flag = 1.

5.70.2.3 min_op_power

```
uint32_t min_op_power
```

Minimum operating power in 250mW units.

5.70.2.4 no_usb_suspend

```
uint32_t no_usb_suspend
```

No USB suspend.

5.70.2.5 obj_pos

```
uint32_t obj_pos
```

Object position.

5.70.2.6 op_power

uint32_t op_power

Operating power in 250mW units.

5.70.2.7 rsrvd1

uint32_t rsrvd1

Reserved field.

5.70.2.8 rsrvd2

uint32_t rsrvd2

Reserved field.

5.70.2.9 unchunk_sup

uint32_t unchunk_sup

Unchunked extended messages supported.

5.70.2.10 usb_comm_cap

uint32_t usb_comm_cap

USB communication capability.

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

- [pd_common/pd.h](#)

5.71 pd_do_t::RDO_FIXED_VAR Struct Reference

```
#include <pd.h>
```

Data Fields

- uint32_t [max_op_current](#): 10
- uint32_t [op_current](#): 10
- uint32_t [rsrvd1](#): 3
- uint32_t [unchunk_sup](#): 1
- uint32_t [no_usb_suspend](#): 1
- uint32_t [usb_comm_cap](#): 1
- uint32_t [cap_mismatch](#): 1
- uint32_t [give_back_flag](#): 1
- uint32_t [obj_pos](#): 3
- uint32_t [rsrvd2](#): 1

5.71.1 Detailed Description

Structure representing a Fixed or Variable Request Data Object.

5.71.2 Field Documentation

5.71.2.1 cap_mismatch

uint32_t cap_mismatch

Capability mismatch.

5.71.2.2 give_back_flag

uint32_t give_back_flag

GiveBack flag = 0.

5.71.2.3 max_op_current

uint32_t max_op_current

Maximum operating current in 10mA units.

5.71.2.4 no_usb_suspend

uint32_t no_usb_suspend

No USB suspend.

5.71.2.5 obj_pos

uint32_t obj_pos

Object position.

5.71.2.6 op_current

uint32_t op_current

Operating current in 10mA units.

5.71.2.7 rsrvd1

uint32_t rsrvd1

Reserved field.

5.71.2.8 rsrvd2

uint32_t rsrvd2

Reserved field.

5.71.2.9 unchunk_sup

uint32_t unchunk_sup

Unchunked extended messages supported.

5.71.2.10 usb_comm_cap

uint32_t usb_comm_cap

USB communication capability.

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

- [pd_common/pd.h](#)

5.72 pd_do_t::RDO_FIXED_VAR_GIVEBACK Struct Reference

```
#include <pd.h>
```

Data Fields

- uint32_t [min_op_current](#): 10
- uint32_t [op_current](#): 10
- uint32_t [rsrvd1](#): 3
- uint32_t [unchunk_sup](#): 1
- uint32_t [no_usb_suspend](#): 1
- uint32_t [usb_comm_cap](#): 1
- uint32_t [cap_mismatch](#): 1
- uint32_t [give_back_flag](#): 1
- uint32_t [obj_pos](#): 3
- uint32_t [rsrvd2](#): 1

5.72.1 Detailed Description

Structure representing a Fixed or Variable Request Data Object with GiveBack.

5.72.2 Field Documentation

5.72.2.1 cap_mismatch

uint32_t cap_mismatch

Capability mismatch.

5.72.2.2 give_back_flag

uint32_t give_back_flag

GiveBack flag = 1.

5.72.2.3 min_op_current

uint32_t min_op_current

Minimum operating current in 10mA units.

5.72.2.4 no_usb_suspend

uint32_t no_usb_suspend

No USB suspend.

5.72.2.5 obj_pos

uint32_t obj_pos

Object position.

5.72.2.6 op_current

uint32_t op_current

Operating current in 10mA units.

5.72.2.7 rsrvd1

uint32_t rsrvd1

Reserved field.

5.72.2.8 rsrvd2

uint32_t rsrvd2

Reserved field.

5.72.2.9 unchunk_sup

uint32_t unchunk_sup

Unchunked extended messages supported.

5.72.2.10 usb_comm_cap

uint32_t usb_comm_cap

USB communication capability.

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

- [pd_common/pd.h](#)

5.73 pd_do_t::RDO_GEN Struct Reference

```
#include <pd.h>
```


Data Fields

- uint32_t `min_max_power_cur`: 10
- uint32_t `op_power_cur`: 10
- uint32_t `rsrvd1`: 3
- uint32_t `unchunk_sup`: 1
- uint32_t `no_usb_suspend`: 1
- uint32_t `usb_comm_cap`: 1
- uint32_t `cap_mismatch`: 1
- uint32_t `give_back_flag`: 1
- uint32_t `obj_pos`: 3
- uint32_t `rsrvd2`: 1

5.73.1 Detailed Description

Structure representing a generic Request Data Object.

5.73.2 Field Documentation

5.73.2.1 cap_mismatch

uint32_t cap_mismatch

Capability mismatch.

5.73.2.2 give_back_flag

uint32_t give_back_flag

GiveBack supported flag = 0.

5.73.2.3 min_max_power_cur

uint32_t min_max_power_cur

Min/Max power or current requirement.

5.73.2.4 no_usb_suspend

uint32_t no_usb_suspend

No USB suspend.

5.73.2.5 obj_pos

uint32_t obj_pos

Object position.

5.73.2.6 op_power_cur

uint32_t op_power_cur

Operating power or current requirement.

5.73.2.7 rsrvd1

uint32_t rsrvd1

Reserved field.

5.73.2.8 rsrvd2

uint32_t rsrvd2

Reserved field.

5.73.2.9 unchunk_sup

uint32_t unchunk_sup

Unchunked extended messages supported.

5.73.2.10 usb_comm_cap

uint32_t usb_comm_cap

USB communication capability.

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

- [pd_common/pd.h](#)

5.74 pd_do_t::RDO_GEN_GVB Struct Reference

```
#include <pd.h>
```

Data Fields

- uint32_t [max_power_cur](#): 10
- uint32_t [op_power_cur](#): 10
- uint32_t [rsrvd1](#): 3
- uint32_t [unchunk_sup](#): 1
- uint32_t [no_usb_suspend](#): 1
- uint32_t [usb_comm_cap](#): 1
- uint32_t [cap_mismatch](#): 1
- uint32_t [give_back_flag](#): 1
- uint32_t [obj_pos](#): 3
- uint32_t [rsrvd2](#): 1

5.74.1 Detailed Description

Structure representing a Generic Request Data Object with GiveBack.

5.74.2 Field Documentation

5.74.2.1 cap_mismatch

uint32_t cap_mismatch

Capability mismatch.

5.74.2.2 give_back_flag

uint32_t give_back_flag

GiveBack supported flag = 1.

5.74.2.3 max_power_cur

uint32_t max_power_cur

Min/Max power or current requirement.

5.74.2.4 no_usb_suspend

uint32_t no_usb_suspend

No USB suspend.

5.74.2.5 obj_pos

uint32_t obj_pos

Object position.

5.74.2.6 op_power_cur

uint32_t op_power_cur

Operating power or current requirement.

5.74.2.7 rsrvd1

uint32_t rsrvd1

Reserved field.

5.74.2.8 rsrvd2

uint32_t rsrvd2

Reserved field.

5.74.2.9 unchunk_sup

uint32_t unchunk_sup

Unchunked extended messages supported.

5.74.2.10 usb_comm_cap

uint32_t usb_comm_cap

USB communication capability.

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

- [pd_common/pd.h](#)

5.75 pd_do_t::RDO_PPS Struct Reference

```
#include <pd.h>
```

Data Fields

- uint32_t [op_cur](#): 7
- uint32_t [rsvd1](#): 2
- uint32_t [out_volt](#): 11
- uint32_t [rsvd2](#): 3
- uint32_t [unchunk_sup](#): 1
- uint32_t [no_usb_suspend](#): 1
- uint32_t [usb_comm_cap](#): 1
- uint32_t [cap_mismatch](#): 1
- uint32_t [rsvd3](#): 1
- uint32_t [obj_pos](#): 3
- uint32_t [rsvd4](#): 1

5.75.1 Detailed Description

Programmable Request Data Object.

5.75.2 Field Documentation

5.75.2.1 cap_mismatch

uint32_t cap_mismatch

Capability mismatch flag.

5.75.2.2 no_usb_suspend

uint32_t no_usb_suspend

No USB suspend flag.

5.75.2.3 obj_pos

uint32_t obj_pos

Object position - index to source PDO.

5.75.2.4 op_cur

uint32_t op_cur

Operating current in 50 mA units.

5.75.2.5 out_volt

uint32_t out_volt

Requested output voltage in 20 mV units.

5.75.2.6 rsvd1

uint32_t rsvd1

Reserved field.

5.75.2.7 rsvd2

uint32_t rsvd2

Reserved field.

5.75.2.8 rsvd3

uint32_t rsvd3

Reserved field.

5.75.2.9 rsvd4

uint32_t rsvd4

Reserved field.

5.75.2.10 unchunk_sup

uint32_t unchunk_sup

Whether unchunked extended messages are supported.

5.75.2.11 usb_comm_cap

uint32_t usb_comm_cap

Whether sink supports USB communication.

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

- [pd_common/pd.h](#)

5.76 `reg_am_t` Struct Reference

```
#include <alt_modes_mngr.h>
```

Data Fields

- [uint16_t svid](#)
- [reg_alt_modes reg_am_ptr](#)

5.76.1 Detailed Description

structure to hold the alternate modes SVID and handler.

5.76.2 Field Documentation

5.76.2.1 `reg_am_ptr`

```
reg_alt_modes reg_am_ptr
```

Alternate mode SVID handler.

5.76.2.2 `svid`

```
uint16_t svid
```

Alternate mode SVID.

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

- [app/alt_mode/alt_modes_mngr.h](#)

5.77 `scp_settings_t` Struct Reference

```
#include <pd.h>
```

Data Fields

- [uint8_t table_len](#)
- [uint8_t threshold](#)
- [uint8_t debounce](#)
- [uint8_t retry_cnt](#)

5.77.1 Detailed Description

Struct to hold the SCP settings.

5.77.2 Field Documentation

5.77.2.1 debounce

uint8_t debounce

SCP debounce duration in micro-seconds. If a non-zero debounce is specified, there can be an error of upto 35 us due to the device being in sleep mode.

5.77.2.2 retry_cnt

uint8_t retry_cnt

Number of consecutive SCP events allowed before the port operation is suspended.

5.77.2.3 table_len

uint8_t table_len

Table length in bytes

5.77.2.4 threshold

uint8_t threshold

SCP threshold: Reduced voltage below expected value in percentage of expected voltage

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

- [pd_common/pd.h](#)

5.78 sensor_data_t Struct Reference

```
#include <pd.h>
```

Data Fields

- [uint8_t sensor_ctrl](#)
- [uint8_t sensor_oc1](#)
- [uint8_t sensor_oc2](#)
- [uint8_t sensor_oc3](#)

5.78.1 Detailed Description

Struct to hold the sensor throttling settings.

5.78.2 Field Documentation

5.78.2.1 sensor_ctrl

uint8_t sensor_ctrl

Bit 7: 0 -> Disabled, 1 -> Enabled; Bit 6-0: I2C address

5.78.2.2 sensor_oc1

```
uint8_t sensor_oc1
```

Maximum sensor temperature °C for the system to function in OC1 (100%) power rating.

5.78.2.3 sensor_oc2

```
uint8_t sensor_oc2
```

Maximum sensor temperature °C for the system to function in OC2 (50%) power rating. To skip this power level, load with the 0x00. To terminate at this level, load with 0xFF.

5.78.2.4 sensor_oc3

```
uint8_t sensor_oc3
```

Maximum sensor temperature °C for the system to function in OC3 (15W) power rating. To skip this power level, load with the 0x00. To terminate at this level, load with 0xFF. Beyond this threshold, the port shall be shutoff.

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

- [pd_common/pd.h](#)

5.79 pd_do_t::SRC_GEN Struct Reference

```
#include <pd.h>
```

Data Fields

- uint32_t [max_cur_power](#): 10
- uint32_t [min_voltage](#): 10
- uint32_t [max_voltage](#): 10
- uint32_t [supply_type](#): 2

5.79.1 Detailed Description

Structure representing a generic source PDO.

5.79.2 Field Documentation

5.79.2.1 max_cur_power

```
uint32_t max_cur_power
```

Maximum current in 10 mA or power in 250 mW units.

5.79.2.2 max_voltage

```
uint32_t max_voltage
```

Maximum voltage in 50mV units.

5.79.2.3 min_voltage

uint32_t min_voltage

Minimum voltage in 50mV units.

5.79.2.4 supply_type

uint32_t supply_type

Supply type.

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

- [pd_common/pd.h](#)

5.80 pd_do_t::STD_AMA_VDO Struct Reference

```
#include <pd.h>
```

Data Fields

- uint32_t [usb_ss_sup](#): 3
- uint32_t [vbus_req](#): 1
- uint32_t [vcon_req](#): 1
- uint32_t [vcon_pwr](#): 3
- uint32_t [ssrx2](#): 1
- uint32_t [ssrx1](#): 1
- uint32_t [sstx2](#): 1
- uint32_t [sstx1](#): 1
- uint32_t [rsvd1](#): 12
- uint32_t [ama_fw_ver](#): 4
- uint32_t [ama_hw_ver](#): 4

5.80.1 Detailed Description

AMA VDO structure as defined by PD 2.0.

5.80.2 Field Documentation

5.80.2.1 ama_fw_ver

uint32_t ama_fw_ver

AMA firmware version.

5.80.2.2 ama_hw_ver

uint32_t ama_hw_ver

AMA hardware version.

5.80.2.3 rsvd1

uint32_t rsvd1

Reserved field.

5.80.2.4 ssrx1

uint32_t ssrx1

Whether SSRX1 has configurable direction.

5.80.2.5 ssrx2

uint32_t ssrx2

Whether SSRX2 has configurable direction.

5.80.2.6 sstx1

uint32_t sstx1

Whether SSTX1 has configurable direction.

5.80.2.7 sstx2

uint32_t sstx2

Whether SSTX2 has configurable direction.

5.80.2.8 usb_ss_sup

uint32_t usb_ss_sup

USB signalling supported.

5.80.2.9 vbus_req

uint32_t vbus_req

Whether device requires VBus.

5.80.2.10 vcon_pwr

uint32_t vcon_pwr

VConn power required.

5.80.2.11 vcon_req

uint32_t vcon_req

Whether device requires VConn.

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

- [pd_common/pd.h](#)

5.81 pd_do_t::STD_AMA_VDO_PD3 Struct Reference

```
#include <pd.h>
```

Data Fields

- uint32_t [usb_ss_sup](#): 3
- uint32_t [vbus_req](#): 1
- uint32_t [vcon_req](#): 1
- uint32_t [vcon_pwr](#): 3
- uint32_t [rsvd1](#): 13
- uint32_t [vdo_version](#): 3
- uint32_t [ama_fw_ver](#): 4
- uint32_t [ama_hw_ver](#): 4

5.81.1 Detailed Description

AMA VDO structure as defined by PD 3.0.

5.81.2 Field Documentation

5.81.2.1 [ama_fw_ver](#)

```
uint32_t ama_fw_ver
```

AMA firmware version.

5.81.2.2 [ama_hw_ver](#)

```
uint32_t ama_hw_ver
```

AMA hardware version.

5.81.2.3 [rsvd1](#)

```
uint32_t rsvd1
```

Reserved field.

5.81.2.4 [usb_ss_sup](#)

```
uint32_t usb_ss_sup
```

USB signalling supported.

5.81.2.5 [vbus_req](#)

```
uint32_t vbus_req
```

Whether device requires VBus.

5.81.2.6 vcon_pwr

```
uint32_t vcon_pwr
```

VConn power required.

5.81.2.7 vcon_req

```
uint32_t vcon_req
```

Whether device requires VConn.

5.81.2.8 vdo_version

```
uint32_t vdo_version
```

VDO version.

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

- [pd_common/pd.h](#)

5.82 pd_do_t::STD_CBL_VDO Struct Reference

```
#include <pd.h>
```

Data Fields

- uint32_t [usb_ss_sup](#): 3
- uint32_t [sop_dp](#): 1
- uint32_t [vbus_thru_cbl](#): 1
- uint32_t [vbus_cur](#): 2
- uint32_t [ssrx2](#): 1
- uint32_t [ssrx1](#): 1
- uint32_t [sstx2](#): 1
- uint32_t [sstx1](#): 1
- uint32_t [cbl_term](#): 2
- uint32_t [cbl_latency](#): 4
- uint32_t [typec_plug](#): 1
- uint32_t [typec_abc](#): 2
- uint32_t [rsvd1](#): 4
- uint32_t [cbl_fw_ver](#): 4
- uint32_t [cbl_hw_ver](#): 4

5.82.1 Detailed Description

Cable VDO structure as defined in USB-PD r2.0.

5.82.2 Field Documentation

5.82.2.1 cbl_fw_ver`uint32_t cbl_fw_ver`

Cable firmware version.

5.82.2.2 cbl_hw_ver`uint32_t cbl_hw_ver`

Cable hardware version.

5.82.2.3 cbl_latency`uint32_t cbl_latency`

Cable latency.

5.82.2.4 cbl_term`uint32_t cbl_term`

Cable termination and VConn power requirement.

5.82.2.5 rsvd1`uint32_t rsvd1`

Reserved field.

5.82.2.6 sop_dp`uint32_t sop_dp`

Whether SOP" controller is present.

5.82.2.7 ssrx1`uint32_t ssrx1`

Whether SSRX1 has configurable direction.

5.82.2.8 ssrx2`uint32_t ssrx2`

Whether SSRX2 has configurable direction.

5.82.2.9 sstx1`uint32_t sstx1`

Whether SSTX1 has configurable direction.

5.82.2.10 sstx2

```
uint32_t sstx2
```

Whether SSTX2 has configurable direction.

5.82.2.11 typec_abc

```
uint32_t typec_abc
```

Cable plug type.

5.82.2.12 typec_plug

```
uint32_t typec_plug
```

Whether cable has a plug: Should be 0.

5.82.2.13 usb_ss_sup

```
uint32_t usb_ss_sup
```

USB signalling supported by the cable.

5.82.2.14 vbus_cur

```
uint32_t vbus_cur
```

VBus current supported by the cable.

5.82.2.15 vbus_thru_cbl

```
uint32_t vbus_thru_cbl
```

Whether cable allows VBus power through.

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

- [pd_common/pd.h](#)

5.83 pd_do_t::STD_CERT_VDO Struct Reference

```
#include <pd.h>
```

Data Fields

- uint32_t [usb_xid](#): 32

5.83.1 Detailed Description

Cert Stat VDO structure.

5.83.2 Field Documentation

5.83.2.1 usb_xid

uint32_t usb_xid

32-bit XID value.

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

- [pd_common/pd.h](#)

5.84 pd_do_t::STD_DP_VDO Struct Reference

```
#include <pd.h>
```

Data Fields

- uint32_t [port_cap](#): 2
- uint32_t [signal](#): 4
- uint32_t [recep](#): 1
- uint32_t [usb2_0](#): 1
- uint32_t [dfp_d_pin](#): 8
- uint32_t [ufp_d_pin](#): 8
- uint32_t [rsvd](#): 8

5.84.1 Detailed Description

DisplayPort Mode VDO as defined by VESA spec.

5.84.2 Field Documentation

5.84.2.1 dfp_d_pin

uint32_t dfp_d_pin

DFP_D pin assignments supported.

5.84.2.2 port_cap

uint32_t port_cap

Port capability.

5.84.2.3 recep

uint32_t recep

Whether Type-C connector is plug or receptacle.

5.84.2.4 rsvd

uint32_t rsvd

Reserved field.

5.84.2.5 signal

uint32_t signal

Signalling supported.

5.84.2.6 ufp_d_pin

uint32_t ufp_d_pin

UFP_D pin assignments supported.

5.84.2.7 usb2_0

uint32_t usb2_0

USB 2.0 signalling required.

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

- [pd_common/pd.h](#)

5.85 pd_do_t::STD_PROD_VDO Struct Reference

```
#include <pd.h>
```

Data Fields

- uint32_t [bcd_dev](#): 16
- uint32_t [usb_pid](#): 16

5.85.1 Detailed Description

Product VDO structure.

5.85.2 Field Documentation

5.85.2.1 bcd_dev

uint32_t bcd_dev

16-bit bcdDevice value.

5.85.2.2 usb_pid

uint32_t usb_pid

16-bit product ID.

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

- [pd_common/pd.h](#)

5.86 pd_do_t::STD_SVID_RESP_VDO Struct Reference

```
#include <pd.h>
```

Data Fields

- uint32_t [svid_n1](#): 16
- uint32_t [svid_n](#): 16

5.86.1 Detailed Description

Discover_SVID response structure.

5.86.2 Field Documentation

5.86.2.1 svid_n

uint32_t svid_n

SVID #2

5.86.2.2 svid_n1

uint32_t svid_n1

SVID #1

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

- [pd_common/pd.h](#)

5.87 pd_do_t::STD_VDM_HDR Struct Reference

```
#include <pd.h>
```

Data Fields

- uint32_t [cmd](#): 5
- uint32_t [rsvd1](#): 1
- uint32_t [cmd_type](#): 2

- uint32_t `obj_pos`: 3
- uint32_t `rsvd2`: 2
- uint32_t `st_ver`: 2
- uint32_t `vdm_type`: 1
- uint32_t `svid`: 16

5.87.1 Detailed Description

Structure representing a Structured VDM Header Data Object.

5.87.2 Field Documentation

5.87.2.1 `cmd`

uint32_t `cmd`

VDM command id.

5.87.2.2 `cmd_type`

uint32_t `cmd_type`

VDM command type.

5.87.2.3 `obj_pos`

uint32_t `obj_pos`

Object position.

5.87.2.4 `rsvd1`

uint32_t `rsvd1`

Reserved field.

5.87.2.5 `rsvd2`

uint32_t `rsvd2`

Reserved field.

5.87.2.6 `st_ver`

uint32_t `st_ver`

Structured VDM version.

5.87.2.7 `svid`

uint32_t `svid`

SVID associated with VDM.

5.87.2.8 vdm_type

uint32_t vdm_type

VDM type = Structured.

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

- [pd_common/pd.h](#)

5.88 pd_do_t::STD_VDM_ID_HDR Struct Reference

```
#include <pd.h>
```

Data Fields

- uint32_t [usb_vid](#): 16
- uint32_t [rsvd1](#): 7
- uint32_t [prod_type_dfp](#): 3
- uint32_t [mod_support](#): 1
- uint32_t [prod_type](#): 3
- uint32_t [usb_dev](#): 1
- uint32_t [usb_host](#): 1

5.88.1 Detailed Description

Structure representing a Standard ID_HEADER VDO.

5.88.2 Field Documentation

5.88.2.1 mod_support

uint32_t mod_support

Whether alternate modes are supported.

5.88.2.2 prod_type

uint32_t prod_type

Product type as UFP.

5.88.2.3 prod_type_dfp

uint32_t prod_type_dfp

Product type as DFP.

5.88.2.4 rsvd1

uint32_t rsvd1

Reserved field.

5.88.2.5 usb_dev

uint32_t usb_dev

USB device supported.

5.88.2.6 usb_host

uint32_t usb_host

USB host supported.

5.88.2.7 usb_vid

uint32_t usb_vid

16-bit vendor ID.

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

- [pd_common/pd.h](#)

5.89 sys_fw_metadata_t Struct Reference

```
#include <boot.h>
```

Data Fields

- [uint8_t fw_checksum](#)
- [uint32_t fw_entry](#)
- [uint16_t boot_last_row](#)
- [uint8_t reserved1 \[2\]](#)
- [uint32_t fw_size](#)
- [uint8_t reserved2 \[3\]](#)
- [uint8_t active_boot_app](#)
- [uint8_t boot_app_ver_status](#)
- [uint16_t boot_app_version](#)
- [uint16_t boot_app_id](#)
- [uint16_t metadata_valid](#)
- [uint32_t fw_version](#)
- [uint32_t boot_seq](#)

5.89.1 Detailed Description

CCGx Firmware metadata structure.

This structure defines the format of the firmware metadata that is stored on device flash. The boot-loader uses the metadata to identify the firmware validity, location, size, start address etc. The metadata for the two runtime firmware images (FW1 and FW2) are located at fixed addresses (for each CCGx part), allowing the boot-loader to precisely locate and validate the flash content during boot-up.

5.89.2 Field Documentation

5.89.2.1 active_boot_app

uint8_t active_boot_app

Offset 10: Creator specific field. Not used in this implementation.

5.89.2.2 boot_app_id

uint16_t boot_app_id

Offset 14: Creator specific field. Not used in this implementation.

5.89.2.3 boot_app_ver_status

uint8_t boot_app_ver_status

Offset 11: Creator specific field. Not used in this implementation.

5.89.2.4 boot_app_version

uint16_t boot_app_version

Offset 12: Creator specific field. Not used in this implementation.

5.89.2.5 boot_last_row

uint16_t boot_last_row

Offset 05: Last Flash row of Bootloader or previous firmware.

5.89.2.6 boot_seq

uint32_t boot_seq

Offset 1C: Boot sequence number field. Boot-loader will load the valid FW copy that has the higher sequence number associated with it.

5.89.2.7 fw_checksum

uint8_t fw_checksum

Offset 00: Single Byte FW Checksum.

5.89.2.8 fw_entry

uint32_t fw_entry

Offset 01: FW Entry Address

5.89.2.9 fw_size

uint32_t fw_size

Offset 09: Size of Firmware.

5.89.2.10 fw_version

uint32_t fw_version

Offset 18: Creator specific field. Not used in this implementation.

5.89.2.11 metadata_valid

uint16_t metadata_valid

Offset 16: Metadata Valid field. Valid if contains "CY".

5.89.2.12 reserved1

uint8_t reserved1[2]

Offset 07: Reserved.

5.89.2.13 reserved2

uint8_t reserved2[3]

Offset 0D: Reserved.

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

- [system/boot.h](#)

5.90 pd_do_t::TBT_CBL_VDO Struct Reference

```
#include <pd.h>
```

Data Fields

- uint32_t [intel_mode](#): 16
- uint32_t [cbl_speed](#): 3
- uint32_t [cbl_gen](#): 2
- uint32_t [cbl_type](#): 1
- uint32_t [b22_retimer_cbl](#): 1
- uint32_t [link_training](#): 1
- uint32_t [rsvd1](#): 8

5.90.1 Detailed Description

Thunderbolt Discover Modes Response Data Object.

5.90.2 Field Documentation

5.90.2.1 b22_retimer_cbl

```
uint32_t b22_retimer_cbl
```

Type of Type-C cable: Redriver or Retimer.

5.90.2.2 cbl_gen

```
uint32_t cbl_gen
```

Thunderbolt cable generation.

5.90.2.3 cbl_speed

```
uint32_t cbl_speed
```

Data bandwidth supported by the Type-C cable.

5.90.2.4 cbl_type

```
uint32_t cbl_type
```

Whether cable is non-optical or optical.

5.90.2.5 intel_mode

```
uint32_t intel_mode
```

Thunderbolt (Intel) modes identifier.

5.90.2.6 link_training

```
uint32_t link_training
```

Type of link training supported by active cable.

5.90.2.7 rsvd1

```
uint32_t rsvd1
```

Reserved field.

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

- [pd_common/pd.h](#)

5.91 pd_do_t::TBT_UFP_VDO Struct Reference

```
#include <pd.h>
```

Data Fields

- uint32_t [intel_mode](#): 16
- uint32_t [adapter](#): 1
- uint32_t [rsvd0](#): 9
- uint32_t [vpro_supp](#): 1
- uint32_t [rsvd1](#): 5

5.91.1 Detailed Description

Thunderbolt UFP Discover Modes Response Data Object.

5.91.2 Field Documentation

5.91.2.1 adapter

uint32_t adapter

Legacy TBT Adapter or Device.

5.91.2.2 intel_mode

uint32_t intel_mode

Thunderbolt (Intel) modes identifier.

5.91.2.3 rsvd0

uint32_t rsvd0

Reserved field.

5.91.2.4 rsvd1

uint32_t rsvd1

Reserved field.

5.91.2.5 vpro_supp

uint32_t vpro_supp

Whether supports vPro mode.

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

- [pd_common/pd.h](#)

5.92 pd_do_t::TBT_VDO Struct Reference

```
#include <pd.h>
```


Data Fields

- uint32_t [intel_mode](#): 16
- uint32_t [cbl_speed](#): 3
- uint32_t [cbl_gen](#): 2
- uint32_t [cbl_type](#): 1
- uint32_t [b22_retimer_cbl](#): 1
- uint32_t [link_training](#): 1
- uint32_t [adapter](#): 1
- uint32_t [cable_active](#): 1
- uint32_t [vpro_dock_host](#): 1
- uint32_t [rsvd1](#): 3
- uint32_t [rsvd2](#): 2

5.92.1 Detailed Description

Thunderbolt Discover Modes Response Data Object.

5.92.2 Field Documentation

5.92.2.1 adapter

uint32_t adapter

Legacy TBT Adapter or Device.

5.92.2.2 b22_retimer_cbl

uint32_t b22_retimer_cbl

Type of Type-C cable: Redriver or Retimer.

5.92.2.3 cable_active

uint32_t cable_active

Whether cable reports active or passive in ID HDR in discover id response.

5.92.2.4 cbl_gen

uint32_t cbl_gen

Thunderbolt cable generation.

5.92.2.5 cbl_speed

uint32_t cbl_speed

Data bandwidth supported by the Type-C cable.

5.92.2.6 cbl_type

uint32_t cbl_type

Whether cable is non-optical or optical.

5.92.2.7 intel_mode

uint32_t intel_mode

Thunderbolt (Intel) modes identifier.

5.92.2.8 link_training

uint32_t link_training

Type of link training supported by active cable.

5.92.2.9 rsvd1

uint32_t rsvd1

Reserved field.

5.92.2.10 rsvd2

uint32_t rsvd2

Reserved field.

5.92.2.11 vpro_dock_host

uint32_t vpro_dock_host

Whether the device supports VPRO feature.

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

- [pd_common/pd.h](#)

5.93 tbthost_cfg_settings_t Struct Reference

```
#include <pd.h>
```

Data Fields

- [uint8_t table_len](#)
- [uint8_t tbt_ctrlr_type](#)
- [uint8_t pref_pwr_role](#)
- [uint8_t pref_data_role](#)
- [uint8_t hpd_handling](#)
- [uint8_t vpro_capable](#)
- [uint8_t sbu_config](#)
- [uint8_t usb4_support](#)
- [uint8_t usb3_support](#)

- [uint8_t host_support](#)
- [uint8_t non_tbt_mux](#)
- [uint8_t rsvd0](#)

5.93.1 Detailed Description

Struct to hold Thunderbolt Host related config settings.

5.93.2 Field Documentation

5.93.2.1 host_support

`uint8_t host_support`

Protocol capabilities (TBT, DP, PCIe) of the host controller.

5.93.2.2 hpd_handling

`uint8_t hpd_handling`

Type of HPD handling (GPIO/I2C) used in the design.

5.93.2.3 non_tbt_mux

`uint8_t non_tbt_mux`

Informs that non TBT MUX is used.

5.93.2.4 pref_data_role

`uint8_t pref_data_role`

Preferred data role for the design. Will be enforced through DR_SWAP.

5.93.2.5 pref_pwr_role

`uint8_t pref_pwr_role`

Preferred power role for the design. Will be enforced through PR_SWAP.

5.93.2.6 rsvd0

`uint8_t rsvd0`

Reserved byte for future use.

5.93.2.7 sbu_config

`uint8_t sbu_config`

Type of SBU configuration used in the design.

5.93.2.8 table_len

```
uint8_t table_len
```

Table length in bytes.

5.93.2.9 tbt_ctrlr_type

```
uint8_t tbt_ctrlr_type
```

Type of Thunderbolt Controller used in the design.

5.93.2.10 usb3_support

```
uint8_t usb3_support
```

USB 3.2 roles supported by the host design.

5.93.2.11 usb4_support

```
uint8_t usb4_support
```

USB4 roles supported by the host design.

5.93.2.12 vpro_capable

```
uint8_t vpro_capable
```

Whether the design is capable of working with VPro docks. Requires HPD handling to be I2C based.

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

- [pd_common/pd.h](#)

5.94 pd_do_t::UFP_VDO_1 Struct Reference

```
#include <pd.h>
```

Data Fields

- uint32_t [usb_sig](#): 3
- uint32_t [alt_modes](#): 3
- uint32_t [rsvd0](#): 18
- uint32_t [dev_cap](#): 4
- uint32_t [rsvd1](#): 1
- uint32_t [vdo_version](#): 3

5.94.1 Detailed Description

UFP VDO #1.

5.94.2 Field Documentation

5.94.2.1 alt_modes

uint32_t alt_modes

Alt. modes supported bit-map.

5.94.2.2 dev_cap

uint32_t dev_cap

Device Capability.

5.94.2.3 rsvd0

uint32_t rsvd0

Reserved field.

5.94.2.4 rsvd1

uint32_t rsvd1

Reserved field.

5.94.2.5 usb_sig

uint32_t usb_sig

USB signaling supported.

5.94.2.6 vdo_version

uint32_t vdo_version

VDO version field.

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

- [pd_common/pd.h](#)

5.95 pd_do_t::USTD_QC_4_0_HDR Struct Reference

```
#include <pd.h>
```

Data Fields

- uint32_t [cmd_0](#): 8
- uint32_t [cmd_1](#): 7
- uint32_t [vdm_type](#): 1
- uint32_t [svid](#): 16

5.95.1 Detailed Description

Structure representing an Unstructured VDM header data object as defined by QC 4.0 spec.

5.95.2 Field Documentation

5.95.2.1 cmd_0

uint32_t cmd_0

Command code #0.

5.95.2.2 cmd_1

uint32_t cmd_1

Command code #1.

5.95.2.3 svid

uint32_t svid

SVID associated with message.

5.95.2.4 vdm_type

uint32_t vdm_type

VDM type = Unstructured.

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

- [pd_common/pd.h](#)

5.96 pd_do_t::USTD_VDM_HDR Struct Reference

```
#include <pd.h>
```

Data Fields

- uint32_t [cmd](#): 5
- uint32_t [seq_num](#): 3
- uint32_t [rsvd1](#): 3
- uint32_t [cmd_type](#): 2
- uint32_t [vdm_ver](#): 2
- uint32_t [vdm_type](#): 1
- uint32_t [svid](#): 16

5.96.1 Detailed Description

Structure representing an Unstructured VDM header data object as defined by Cypress.

5.96.2 Field Documentation

5.96.2.1 cmd

uint32_t cmd

Command id.

5.96.2.2 cmd_type

uint32_t cmd_type

Command type.

5.96.2.3 rsvd1

uint32_t rsvd1

Reserved field.

5.96.2.4 seq_num

uint32_t seq_num

Sequence number.

5.96.2.5 svid

uint32_t svid

SVID associated with VDM.

5.96.2.6 vdm_type

uint32_t vdm_type

VDM type = Unstructured.

5.96.2.7 vdm_ver

uint32_t vdm_ver

VDM version.

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

- [pd_common/pd.h](#)

5.97 uvp_settings_t Struct Reference

```
#include <pd.h>
```

Data Fields

- uint8_t [table_len](#)
- uint8_t [threshold](#)
- uint8_t [debounce](#)
- uint8_t [retry_cnt](#)

5.97.1 Detailed Description

Struct to hold the UVP settings.

5.97.2 Field Documentation

5.97.2.1 debounce

```
uint8_t debounce
```

UVP debounce duration in micro-seconds. If a non-zero debounce is specified, there can be an error of upto 35 us due to the device being in sleep mode.

5.97.2.2 retry_cnt

```
uint8_t retry_cnt
```

Number of consecutive UVP events allowed before the port operation is suspended by CCG firmware.

5.97.2.3 table_len

```
uint8_t table_len
```

Table length in bytes

5.97.2.4 threshold

```
uint8_t threshold
```

UVP threshold: Reduced voltage below expected value in percentage of expected voltage

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

- [pd_common/pd.h](#)

5.98 pd_do_t::VAR_SNK Struct Reference

```
#include <pd.h>
```

Data Fields

- uint32_t [op_current](#): 10
- uint32_t [min_voltage](#): 10
- uint32_t [max_voltage](#): 10
- uint32_t [supply_type](#): 2

5.98.1 Detailed Description

Structure representing a Variable Supply PDO - Sink.

5.98.2 Field Documentation

5.98.2.1 max_voltage

uint32_t max_voltage

Maximum voltage in 50mV units.

5.98.2.2 min_voltage

uint32_t min_voltage

Minimum voltage in 50mV units.

5.98.2.3 op_current

uint32_t op_current

Operational current in 10mA units.

5.98.2.4 supply_type

uint32_t supply_type

Supply type - should be 'b10.

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

- [pd_common/pd.h](#)

5.99 pd_do_t::VAR_SRC Struct Reference

```
#include <pd.h>
```

Data Fields

- uint32_t [max_current](#): 10
- uint32_t [min_voltage](#): 10
- uint32_t [max_voltage](#): 10
- uint32_t [supply_type](#): 2

5.99.1 Detailed Description

Structure representing a Variable Supply PDO - Source.

5.99.2 Field Documentation

5.99.2.1 max_current

uint32_t max_current

Maximum current in 10mA units.

5.99.2.2 max_voltage

uint32_t max_voltage

Maximum voltage in 50mV units.

5.99.2.3 min_voltage

uint32_t min_voltage

Minimum voltage in 50mV units.

5.99.2.4 supply_type

uint32_t supply_type

Supply type - should be 'b10.

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

- [pd_common/pd.h](#)

5.100 vconn_ocp_settings_t Struct Reference

```
#include <pd.h>
```

Data Fields

- [uint8_t table_len](#)
- [uint8_t threshold](#)
- [uint8_t debounce](#)
- [uint8_t retry_cnt](#)

5.100.1 Detailed Description

Struct to hold the Vconn OCP settings.

5.100.2 Field Documentation

5.100.2.1 debounce

uint8_t debounce

Vconn OCP debounce period in ms.

5.100.2.2 retry_cnt

uint8_t retry_cnt

Number of consecutive OCP events allowed before the port is suspended by CCG firmware.

5.100.2.3 table_len

uint8_t table_len

Table length in bytes

5.100.2.4 threshold

uint8_t threshold

Max. Vconn current allowed in 10 mA units.

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

- [pd_common/pd.h](#)

5.101 vdm_msg_info_t Struct Reference

```
#include <vdm_task_mngr.h>
```

Data Fields

- [pd_do_t vdm_header](#)
- [uint8_t sop_type](#)
- [uint8_t vdo_num](#)
- [pd_do_t vdo](#) [MAX_NO_OF_VDO]

5.101.1 Detailed Description

This struct holds received/sent VDM information which is used by VDM alternative modes managers.

5.101.2 Field Documentation

5.101.2.1 sop_type

uint8_t sop_type

VDM SOP type.

5.101.2.2 vdm_header

[pd_do_t](#) vdm_header

Holds VDM buffer.

5.101.2.3 vdo

`pd_do_t vdo [MAX_NO_OF_VDO]`

VDO objects buffer.

5.101.2.4 vdo_numb

`uint8_t vdo_numb`

Number of received VDOs in VDM.

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

- [app/alt_mode/vdm_task_mngr.h](#)

5.102 vdm_resp_t Struct Reference

```
#include <pd.h>
```

Data Fields

- [pd_do_t resp_buf \[MAX_NO_OF_DO\]](#)
- [uint8_t do_count](#)
- [vdm_ams_t no_resp](#)

5.102.1 Detailed Description

Struct to hold response to policy manager.

5.102.2 Field Documentation

5.102.2.1 do_count

`uint8_t do_count`

Data objects count

5.102.2.2 no_resp

`vdm_ams_t no_resp`

Response type.

5.102.2.3 resp_buf

`pd_do_t resp_buf [MAX_NO_OF_DO]`

Data objects buffer

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

- [pd_common/pd.h](#)

Chapter 6

File Documentation

6.1 app/alt_mode/alt_mode_hw.h File Reference

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

Data Structures

- union [alt_mode_hw_evt_t](#)
- struct [alt_mode_hw_evt_t::ALT_MODE_HW_EVT](#)

Macros

- #define [NO_DATA](#) (0u)
- #define [HPD_ENABLE_CMD](#) (0u)
- #define [HPD_DISABLE_CMD](#) (5u)

Typedefs

- typedef void(* [alt_mode_hw_cmd_cbk_t](#)) (uint8_t port, uint32_t command)

Enumerations

- enum [alt_mode_hw_t](#) { [ALT_MODE_MUX](#) = 1, [ALT_MODE_HPD](#) }
- enum [mux_select_t](#) {
[MUX_CONFIG_ISOLATE](#), [MUX_CONFIG_SAFE](#), [MUX_CONFIG_SS_ONLY](#), [MUX_CONFIG_DP_2_LANE](#),
[MUX_CONFIG_DP_4_LANE](#), [MUX_CONFIG_USB4_CUSTOM](#), [MUX_CONFIG_RIDGE_CUSTOM](#),
[MUX_CONFIG_INIT](#),
[MUX_CONFIG_DEINIT](#) }
- enum [mux_poll_status_t](#) { [MUX_STATE_IDLE](#), [MUX_STATE_FAIL](#), [MUX_STATE_BUSY](#), [MUX_STATE_SUCCESS](#) }

Functions

- void [alt_mode_hw_set_cbk](#) (uint8_t port, [alt_mode_hw_cmd_cbk_t](#) cbk)
- void [alt_mode_hw_deinit](#) (uint8_t port)

- bool [eval_app_alt_hw_cmd](#) (uint8_t port, uint8_t *cmd_param)
- bool [eval_hpd_cmd](#) (uint8_t port, uint32_t cmd)
- bool [eval_mux_cmd](#) (uint8_t port, uint32_t cmd)
- bool [set_mux](#) (uint8_t port, [mux_select_t](#) cfg, uint32_t custom_data)
- void [ignore_mux_changes](#) (uint8_t port, bool ignore)
- [mux_select_t](#) [get_mux_state](#) (uint8_t port)
- bool [alt_mode_hw_is_idle](#) (uint8_t port)
- void [alt_mode_hw_sleep](#) (uint8_t port)
- void [alt_mode_hw_wakeup](#) (uint8_t port)
- bool [dp_snk_get_hpd_state](#) (uint8_t port)

6.1.1 Detailed Description

Hardware control for Alternate mode implementation.

6.1.2 Macro Definition Documentation

6.1.2.1 HPD_DISABLE_CMD

```
#define HPD_DISABLE_CMD (5u)
```

Disable HPD application command value.

6.1.2.2 HPD_ENABLE_CMD

```
#define HPD_ENABLE_CMD (0u)
```

Enable HPD application command value.

6.1.2.3 NO_DATA

```
#define NO_DATA (0u)
```

Zero data.

6.1.3 Enumeration Type Documentation

6.1.3.1 alt_mode_hw_t

```
enum alt_mode_hw_t
```

Application HW type values which are used in [alt_mode_hw_evt_t](#) structure.

Enumerator

ALT_MODE_MUX	HW type - MUX.
ALT_MODE_HPD	HW type - HPD transceiver/receiver.

6.1.3.2 mux_poll_status_t

enum `mux_poll_status_t`

@ typedef `mux_status_t` @ brief Possible states for the MUX handler.

Enumerator

<code>MUX_STATE_IDLE</code>	MUX idle state.
<code>MUX_STATE_FAIL</code>	MUX switch failed.
<code>MUX_STATE_BUSY</code>	MUX is busy.
<code>MUX_STATE_SUCCESS</code>	MUX switched successfully.

6.1.3.3 mux_select_t

enum `mux_select_t`

@ typedef `mux_select_t` @ brief Possible settings for the Type-C Data MUX. @ note This type should be extended to cover all possible modes for the MUX.

Enumerator

<code>MUX_CONFIG_ISOLATE</code>	Isolate configuration.
<code>MUX_CONFIG_SAFE</code>	USB Safe State (USB 2.0 lines remain active)
<code>MUX_CONFIG_SS_ONLY</code>	USB SS configuration.
<code>MUX_CONFIG_DP_2_LANE</code>	Two lane DP configuration.
<code>MUX_CONFIG_DP_4_LANE</code>	Four lane DP configuration.
<code>MUX_CONFIG_USB4_CUSTOM</code>	USB4 custom configuration.
<code>MUX_CONFIG_RIDGE_CUSTOM</code>	Alpine/Titan Ridge custom configuration.
<code>MUX_CONFIG_INIT</code>	Enables MUX functionality.
<code>MUX_CONFIG_DEINIT</code>	Disables MUX functionality.

6.1.4 Function Documentation

6.1.4.1 alt_mode_hw_deinit()

```
void alt_mode_hw_deinit (
    uint8_t port )
```

This function deinit's all HW related to alt modes.

Parameters

<i>port</i>	Port index the function is performed for.
-------------	---

Returns

None.

6.1.4.2 alt_mode_hw_is_idle()

```
bool alt_mode_hw_is_idle (
    uint8_t port )
```

Check whether the ALT. MODE hardware block is idle.

This function is part of the deep-sleep entry checks for the CCG device, and checks whether there are any pending ALT. MODE hardware commands that require the device to be active.

Parameters

<i>port</i>	Port on which the hardware status is to be checked.
-------------	---

Returns

true if the block is idle, false otherwise.

6.1.4.3 alt_mode_hw_set_cbk()

```
void alt_mode_hw_set_cbk (
    uint8_t port,
    alt_mode_hw_cmd_cbk_t cbk )
```

This function registers an ALT. MODE hardware callback function.

Parameters

<i>port</i>	Port for which the callback is registered.
<i>cbk</i>	Callback function for ALT. MODE hardware events.

Returns

None.

6.1.4.4 alt_mode_hw_sleep()

```
void alt_mode_hw_sleep (
    uint8_t port )
```

Prepare the Alt. Mode hardware state for device deep-sleep.

Parameters

<i>port</i>	The port whose hardware state is to be configured.
-------------	--

Returns

None.

6.1.4.5 alt_mode_hw_wakeup()

```
void alt_mode_hw_wakeup (
    uint8_t port )
```

Restore Alt. Mode hardware state after device resumes from deep-sleep.

Parameters

<i>port</i>	The port whose hardware state is to be restored.
-------------	--

Returns

None

6.1.4.6 dp_snk_get_hpd_state()

```
bool dp_snk_get_hpd_state (
    uint8_t port )
```

Return HPD state based on HPD Queue events.

Parameters

<i>port</i>	DP port index
-------------	---------------

Returns

true if HPD is connected, false otherwise

6.1.4.7 eval_app_alt_hw_cmd()

```
bool eval_app_alt_hw_cmd (
    uint8_t port,
    uint8_t * cmd_param )
```

This function evaluates received application HW command.

Parameters

<i>port</i>	Port index the function is performed for.
<i>cmd_param</i>	Pointer to received application HW command data.

Returns

true if APP command passed successful, false if APP command is invalid or contain unacceptable fields.

6.1.4.8 eval_hpd_cmd()

```
bool eval_hpd_cmd (
    uint8_t port,
    uint32_t cmd )
```

This function evaluates received HPD application command.

Parameters

<i>port</i>	Port index the function is performed for.
<i>cmd</i>	Pointer to received HPD application command data.

Returns

true if HPD APP command passed successful, false if HPD APP command is invalid or contain unacceptable fields.

6.1.4.9 eval_mux_cmd()

```
bool eval_mux_cmd (
    uint8_t port,
    uint32_t cmd )
```

This function evaluates received MUX application command.

Parameters

<i>port</i>	Port index the function is performed for.
<i>cmd</i>	Pointer to received MUX application command data.

Returns

true if MUX APP command passed successful, false if MUX APP command is invalid or contain unacceptable fields.

6.1.4.10 get_mux_state()

```
mux_select_t get_mux_state (
    uint8_t port )
```

Get the current state of the MUX.

Parameters

<i>port</i>	PD port index.
-------------	----------------

Returns

Active MUX setting.

6.1.4.11 ignore_mux_changes()

```
void ignore_mux_changes (
    uint8_t port,
    bool ignore )
```

Ignore changes to MUX settings.

Parameters

<i>port</i>	Port index the function is performed for.
<i>ignore</i>	Ignore changes to any MUX state on this

Returns

None

6.1.4.12 set_mux()

```
bool set_mux (
    uint8_t port,
    mux_select_t cfg,
    uint32_t custom_data )
```

This function set appropriate MUX configuration based on the function parameters.

Parameters

<i>port</i>	Port index the function is performed for.
<i>cfg</i>	Required MUX configuration.
<i>custom_data</i>	Additional data in case of custom AR MUX configuration.

Returns

true if MUX set passed successful, false if MUX setting is invalid or contain unacceptable fields.

6.2 app/alt_mode/alt_modes_mgr.h File Reference

```
#include <stdint.h>
#include <pd.h>
#include <config.h>
#include <vdm_task_mgr.h>
```

Data Structures

- union [alt_mode_evt_t](#)
- struct [alt_mode_evt_t::ALT_MODE_EVT](#)
- struct [alt_mode_evt_t::ALT_MODE_EVT_DATA](#)
- struct [alt_mode_reg_info_t](#)
- struct [alt_mode_info_t](#)
- struct [reg_am_t](#)

Macros

- #define [ATCH_TGT](#) (1u)
- #define [CABLE](#) (2u)
- #define [MODE_NOT_SUPPORTED](#) (0xFFu)
- #define [EMPTY_VDO](#) (0x00000000u)
- #define [NONE_VDO](#) (0xFFFFFFFFu)
- #define [MAX_SUPP_ALT_MODES](#) (4u)
- #define [VDO_START_IDX](#) (1u)
- #define [NONE_MODE_MASK](#) (0x00000000u)
- #define [FULL_MASK](#) (0xFFFFFFFFu)
- #define [EN_FLAG_MASK](#) (0x00000001u)
- #define [EXIT_ALL_MODES](#) (0x7u)
- #define [CBL_DIR_SUPP_MASK](#) (0x780u)
- #define [ALT_MODE_EVT_SIZE](#) (2u)
- #define [ALT_MODE_EVT_IDX](#) (0u)
- #define [ALT_MODE_EVT_DATA_IDX](#) (1u)
- #define [NO_DATA](#) (0u)
- #define [MAX_RETRY_CNT](#) (9u)
- #define [AM_SVID_CONFIG_SIZE_IDX](#) (0u)
- #define [AM_SVID_CONFIG_OFFSET_IDX](#) (1u)
- #define [DFP_ALT_MODE_HPI_OFFSET](#) (8u)
- #define [UFP_ALT_MODE_HPI_MASK](#) (0xFFu)
- #define [SET_FLAG](#)(status, bit_idx) ((status) |= (1 << ((uint32_t)(bit_idx))))
- #define [REMOVE_FLAG](#)(status, bit_idx) ((status) &= (~(1 << ((uint32_t)(bit_idx)))))
- #define [IS_FLAG_CHECKED](#)(status, bit_idx) (((status) >> (uint32_t)(bit_idx)) & [EN_FLAG_MASK](#))
- #define [VDM_HDR](#) vdm_header.std_vdm_hdr

Typedefs

- typedef void(* [alt_mode_cbk_t](#)) (uint8_t port)
- typedef bool(* [alt_mode_app_cbk_t](#)) (uint8_t port, [alt_mode_evt_t](#) app_cmd)
- typedef [alt_mode_info_t](#) *(* [reg_alt_modes](#)) (uint8_t port, [alt_mode_reg_info_t](#) *reg_info)

Enumerations

- enum [alt_mode_mgr_state_t](#) { [ALT_MODE_MNGR_STATE_DISC_MODE](#) = 0, [ALT_MODE_MNGR_STATE_PROCESS](#) }
- enum [alt_mode_state_t](#) { [ALT_MODE_STATE_DISABLE](#) = 0, [ALT_MODE_STATE_IDLE](#), [ALT_MODE_STATE_INIT](#), [ALT_MODE_STATE_SEND](#), [ALT_MODE_STATE_WAIT_FOR_RESP](#), [ALT_MODE_STATE_FAIL](#), [ALT_MODE_STATE_RUN](#), [ALT_MODE_STATE_EXIT](#) }

- enum `alt_mode_app_evt_t` {
`AM_NO_EVT = 0, AM_EVT_SVID_NOT_FOUND, AM_EVT_ALT_MODE_ENTERED, AM_EVT_ALT_MODE_EXITED,`
`AM_EVT_DISC_FINISHED, AM_EVT_SVID_NOT_SUPP, AM_EVT_SVID_SUPP, AM_EVT_ALT_MODE_SUPP,`
`AM_EVT_SOP_RESP_FAILED, AM_EVT_CBL_RESP_FAILED, AM_EVT_CBL_NOT_SUPP_ALT_MODE,`
`AM_EVT_NOT_SUPP_PARTNER_CAP,`
`AM_EVT_DATA_EVT, AM_EVT_SUPP_ALT_MODE_CHNG }`
- enum `alt_mode_app_cmd_t` {
`AM_NO_CMD = 0, AM_SET_TRIGGER_MASK, AM_CMD_ENTER = 3, AM_CMD_EXIT,`
`AM_CMD_SPEC, AM_CMD_RUN_UFP_DISC }`
- enum `fail_status_t` { `BUSY = 0, GOOD_CRC_NOT_RSVD, TIMEOUT, NACK }`

Functions

- `vdm_task_t reg_alt_mode_mngr` (uint8_t port, `atch_tgt_info_t` *atch_tgt_info, `vdm_msg_info_t` *vdm_msg←_info)
- `vdm_task_t vdm_task_mngr_alt_mode_process` (uint8_t port, `vdm_evt_t` vdm_evt)
- bool `eval_rec_vdm` (uint8_t port, const `pd_packet_t` *vdm)
- const uint32_t * `form_alt_mode_event` (uint8_t port, uint16_t svid, uint8_t am_idx, `alt_mode_app_evt_t` evt, uint32_t data)
- bool `eval_app_alt_mode_cmd` (uint8_t port, uint8_t *cmd, uint8_t *data)
- bool `is_alt_mode_mngr_idle` (uint8_t port)
- void `alt_mode_mngr_sleep` (uint8_t port)
- void `alt_mode_mngr_wakeup` (uint8_t port)
- void `reset_alt_mode_info` (`alt_mode_info_t` *info)
- `dpm_pd_cmd_buf_t` * `get_vdm_buff` (uint8_t port)
- uint8_t `is_svid_supported` (uint16_t svid, uint8_t port)
- uint8_t `alt_mode_get_status` (uint8_t port)
- void `alt_mode_layer_reset` (uint8_t port)
- void `alt_mode_mngr_reset_info` (uint8_t port)
- void `alt_mode_mngr_exit_all` (uint8_t port, bool send_vdm_exit, `pd_cbk_t` exit_all_cbk)
- uint8_t `get_alt_mode_numb` (uint8_t port)
- uint8_t `get_alt_modes_config_svid_idx` (uint8_t port, `port_type_t` type, uint16_t svid)
- void `set_alt_mode_mask` (uint8_t port, uint16_t mask)
- bool `set_custom_svid` (uint8_t port, uint16_t svid)
- uint16_t `get_custom_svid` (uint8_t port)
- uint16_t `get_svid_from_idx` (uint8_t port, uint8_t idx)
- `alt_mode_info_t` * `get_mode_info` (uint8_t port, uint8_t alt_mode_idx)

6.2.1 Detailed Description

Alternate Mode Manager header file.

6.2.2 Macro Definition Documentation

6.2.2.1 ALT_MODE_EVT_DATA_IDX

```
#define ALT_MODE_EVT_DATA_IDX (1u)
```

Index of data for alt mode APP event.

6.2.2.2 ALT_MODE_EVT_IDX

```
#define ALT_MODE_EVT_IDX (0u)
```

Index of alt mode APP event code.

6.2.2.3 ALT_MODE_EVT_SIZE

```
#define ALT_MODE_EVT_SIZE (2u)
```

Size of alt mode APP event data in 4 byte words.

6.2.2.4 AM_SVID_CONFIG_OFFSET_IDX

```
#define AM_SVID_CONFIG_OFFSET_IDX (1u)
```

Offset of SVID in alternate mode configuration packet.

6.2.2.5 AM_SVID_CONFIG_SIZE_IDX

```
#define AM_SVID_CONFIG_SIZE_IDX (0u)
```

Index of size of alternate mode configuration packet.

6.2.2.6 ATCH_TGT

```
#define ATCH_TGT (1u)
```

Internal alt modes manager denotation of attached UFP target.

6.2.2.7 CABLE

```
#define CABLE (2u)
```

Internal alt modes manager denotation of the attached cable.

6.2.2.8 CBL_DIR_SUPP_MASK

```
#define CBL_DIR_SUPP_MASK (0x780u)
```

Mask to find out cable directionality support for cable discovery ID response.

6.2.2.9 DFP_ALT_MODE_HPI_OFFSET

```
#define DFP_ALT_MODE_HPI_OFFSET (8u)
```

Offset to get which DFP alt modes should be enabled .

6.2.2.10 EMPTY_VDO

```
#define EMPTY_VDO (0x00000000u)
```

Internal alt modes manager denotation in case if VDO sending not needed.

6.2.2.11 EN_FLAG_MASK

```
#define EN_FLAG_MASK (0x00000001u)
```

This mask is used by IS_FLAG_CHECKED macro to check if bit is set in some mask.

6.2.2.12 EXIT_ALL_MODES

```
#define EXIT_ALL_MODES (0x7u)
```

Object position corresponding to an Exit All Modes VDM command.

6.2.2.13 FULL_MASK

```
#define FULL_MASK (0xFFFFFFFFu)
```

All bit set full mask.

6.2.2.14 IS_FLAG_CHECKED

```
#define IS_FLAG_CHECKED(  
    status,  
    bit_idx ) (((status) >> (uint32_t)(bit_idx)) & EN_FLAG_MASK)
```

Return true if bit_idx of the status variable is set to 1.

6.2.2.15 MAX_RETRY_CNT

```
#define MAX_RETRY_CNT (9u)
```

Maximum number of VDM send retries in case of no CRC response, timeout or BUSY response.

6.2.2.16 MAX_SUPP_ALT_MODES

```
#define MAX_SUPP_ALT_MODES (4u)
```

Maximum number of alternate modes which alt modes manager could operate in the same time. Setting this to larger values will increase RAM requirement for the projects.

6.2.2.17 MODE_NOT_SUPPORTED

```
#define MODE_NOT_SUPPORTED (0xFFu)
```

Internal alt modes manager denotation of non supported alternate mode.

6.2.2.18 NO_DATA

```
#define NO_DATA (0u)
```

Internal alt modes manager denotation of object without useful data.

6.2.2.19 NONE_MODE_MASK

```
#define NONE_MODE_MASK (0x00000000u)
```

Empty mask.

6.2.2.20 NONE_VDO

```
#define NONE_VDO (0xFFFFFFFFu)
```

Internal alt modes manager denotation of zero VDO.

6.2.2.21 REMOVE_FLAG

```
#define REMOVE_FLAG(  
    status,  
    bit_idx ) ((status) &= (~(1 << ((uint32_t)(bit_idx)))))
```

Set appropriate `bit_idx` to 0 in the status variable.

6.2.2.22 SET_FLAG

```
#define SET_FLAG(  
    status,  
    bit_idx ) ((status) |= (1 << ((uint32_t)(bit_idx))))
```

Set appropriate `bit_idx` to 1 in the status variable.

6.2.2.23 UFP_ALT_MODE_HPI_MASK

```
#define UFP_ALT_MODE_HPI_MASK (0xFFu)
```

Mask to get which UFP alt modes should be enabled .

6.2.2.24 VDM_HDR

```
#define VDM_HDR vdm_header.std_vdm_hdr
```

Quick access to Standard VDM header fields macros.

6.2.2.25 VDO_START_IDX

```
#define VDO_START_IDX (1u)
```

Start index of VDO data in VDM message.

6.2.3 Typedef Documentation

6.2.3.1 alt_mode_app_cbk_t

```
typedef bool(* alt_mode_app_cbk_t) (uint8_t port, alt_mode_evt_t app_cmd)
```

This type of the function is used by alt modes manager to run alternate mode analysis of received APP command.

Parameters

<i>port</i>	Port index the function is performed for.
<i>hpi_cmd</i>	Received APP command data.

Returns

true if APP command passed successful, false if APP command is invalid or contains unacceptable fields.

6.2.3.2 alt_mode_cbk_t

```
typedef void(* alt_mode_cbk_t) (uint8_t port)
```

This type of the function is used by alt modes manager to communicate with any of supported alt modes.

Parameters

<i>port</i>	Port index the function is performed for .
-------------	--

Returns

None.

6.2.4 Enumeration Type Documentation**6.2.4.1 alt_mode_app_cmd_t**

```
enum alt_mode_app_cmd_t
```

This enumeration holds all possible APP command related to Alt modes handling.

Enumerator

AM_NO_CMD	Empty command.
AM_SET_TRIGGER_MASK	Sets trigger mask to prevent auto-entering of the selected alternate modes.
AM_CMD_ENTER	Enter to selected alternate mode.
AM_CMD_EXIT	Exit from selected alternate mode.
AM_CMD_SPEC	Specific alternate EC mode command with data.
AM_CMD_RUN_UFP_DISC	Runs Discover command when CCG is UFP due to PD 3.0 spec .

6.2.4.2 alt_mode_app_evt_t

```
enum alt_mode_app_evt_t
```

This enumeration holds all possible application events related to Alt modes handling.

Enumerator

AM_NO_EVT	Empty event.
AM_EVT_SVID_NOT_FOUND	Sends to EC if UFP doesn't support any SVID.
AM_EVT_ALT_MODE_ENTERED	Alternate mode entered.
AM_EVT_ALT_MODE_EXITED	Alternate mode exited.
AM_EVT_DISC_FINISHED	Discovery process was finished.

Enumerator

AM_EVT_SVID_NOT_SUPP	CCGx doesn't support received SVID.
AM_EVT_SVID_SUPP	CCGx supports received SVID.
AM_EVT_ALT_MODE_SUPP	CCGx supports alternate mode.
AM_EVT_SOP_RESP_FAILED	UFP VDM response failed.
AM_EVT_CBL_RESP_FAILED	Cable response failed.
AM_EVT_CBL_NOT_SUPP_ALT_MODE	Cable capabilities couldn't provide alternate mode handling.
AM_EVT_NOT_SUPP_PARTNER_CAP	CCGx and UFP capabilities not consistent.
AM_EVT_DATA_EVT	Specific alternate mode event with data.
AM_EVT_SUPP_ALT_MODE_CHNG	Same functionality as ALT_MODE_SUPP; new one for UCSI

6.2.4.3 alt_mode_mgr_state_t

```
enum alt_mode_mgr_state_t
```

This enumeration holds all possible Alt modes manager states when DFP.

Enumerator

ALT_MODE_MNGR_STATE_DISC_MODE	Alt modes manager discovery mode state.
ALT_MODE_MNGR_STATE_PROCESS	Alt modes manager alternate modes processing state.

6.2.4.4 alt_mode_state_t

```
enum alt_mode_state_t
```

This enumeration holds all possible states of each alt mode which is handled by Alt modes manager.

Enumerator

ALT_MODE_STATE_DISABLE	State when alternate mode functionality is disabled.
ALT_MODE_STATE_IDLE	State when alternate mode is idle.
ALT_MODE_STATE_INIT	State when alternate mode initiate its functionality.
ALT_MODE_STATE_SEND	State when alternate mode needs to send VDM message.
ALT_MODE_STATE_WAIT_FOR_RESP	State while alternate mode wait for VDM response.
ALT_MODE_STATE_FAIL	State when alternate mode VDM response fails.
ALT_MODE_STATE_RUN	State when alternate mode need to be running.
ALT_MODE_STATE_EXIT	State when alternate mode exits and resets all related variables.

6.2.4.5 fail_status_t

```
enum fail_status_t
```

Enum of the VDM failure response status.

Enumerator

BUSY	Target is BUSY.
GOOD_CRC_NOT_RSVD	Good CRC wasn't received.
TIMEOUT	No response or corrupted packet.
NACK	Sent VDM NACKed.

6.2.5 Function Documentation

6.2.5.1 alt_mode_get_status()

```
uint8_t alt_mode_get_status (
    uint8_t port )
```

Retrieve the current alternate mode status for a PD port.

Parameters

<i>port</i>	PD port to be queried.
-------------	------------------------

Returns

The alternate mode status.

6.2.5.2 alt_mode_layer_reset()

```
void alt_mode_layer_reset (
    uint8_t port )
```

Reset alt modes layer.

Parameters

<i>port</i>	PD port index.
-------------	----------------

Returns

None

6.2.5.3 alt_mode_mngr_exit_all()

```
void alt_mode_mngr_exit_all (
    uint8_t port,
    bool send_vdm_exit,
    pd_cbk_t exit_all_cbk )
```

Exit all active alternate modes.

Parameters

<i>port</i>	PD port index.
<i>send_vdm_exit</i>	Flag which indicates is sending Exit VDM is required during exit all modes procedure.
<i>exit_all_cbk</i>	Callback which will call after exit all alt modes procedure will be finifed.

Returns

None

6.2.5.4 alt_mode_mgr_reset_info()

```
void alt_mode_mgr_reset_info (
    uint8_t port )
```

Reset the Alternate mode manager state.

Parameters

<i>port</i>	PD port index.
-------------	----------------

Returns

None

6.2.5.5 alt_mode_mgr_sleep()

```
void alt_mode_mgr_sleep (
    uint8_t port )
```

Prepare the alt modes manager for device deep sleep.

Parameters

<i>port</i>	Index of USB-PD port to be prepared for sleep.
-------------	--

Returns

None

6.2.5.6 alt_mode_mgr_wakeup()

```
void alt_mode_mgr_wakeup (
    uint8_t port )
```

Restore the alt modes manager state after waking from deep sleep.

Parameters

<i>port</i>	Index of USB-PD port to be restored.
-------------	--------------------------------------

Returns

None

6.2.5.7 eval_app_alt_mode_cmd()

```
bool eval_app_alt_mode_cmd (
    uint8_t port,
    uint8_t * cmd,
    uint8_t * data )
```

This function analyzes, parses and run alternate mode analysis function if received command is specific alt mode command.

Parameters

<i>port</i>	Port index the function is performed for.
<i>cmd</i>	Pointer to received alt mode APP command.
<i>data</i>	Pointer to received alt mode APP command additional data.

Returns

true if APP command passed successful, false if APP command is invalid or contains unacceptable fields.

6.2.5.8 eval_rec_vdm()

```
bool eval_rec_vdm (
    uint8_t port,
    const pd_packet_t * vdm )
```

This function run received VDM analysis if CCG is UFP.

Parameters

<i>port</i>	Port index the function is performed for.
<i>vdm</i>	Pointer to pd packet which contains received VDM.

Returns

true if received VDM could handled successful and VDM response need to be sent with ACK. If received VDM should be NACKed then returns false

6.2.5.9 form_alt_mode_event()

```
const uint32_t* form_alt_mode_event (
```

```

uint8_t port,
uint16_t svid,
uint8_t am_idx,
alt_mode_app_evt_t evt,
uint32_t data )

```

Fill alt mode APP event fields with appropriate values.

Parameters

<i>port</i>	Port index the event is performed for.
<i>svid</i>	SVID of alternate mode which event refers to.
<i>am_idx</i>	Index of alternate mode in compatibility table which event refers to.
<i>evt</i>	Alternate mode APP event.
<i>data</i>	Alternate mode APP event data.

Returns

pointer to the event related data.

6.2.5.10 get_alt_mode_numb()

```

uint8_t get_alt_mode_numb (
    uint8_t port )

```

Get number of the alternate modes for choosen port.

Parameters

<i>port</i>	PD port index.
-------------	----------------

Returns

Number of the alternates modes for the choosen port and current data role.

6.2.5.11 get_alt_modes_config_svid_idx()

```

uint8_t get_alt_modes_config_svid_idx (
    uint8_t port,
    port_type_t type,
    uint16_t svid )

```

Get index of selected SVID from config table.

Parameters

<i>port</i>	PD port index.
<i>type</i>	type of the port i.e. port_type_t.
<i>svid</i>	SVID.

Returns

SVID index.

6.2.5.12 get_custom_svid()

```
uint16_t get_custom_svid (
    uint8_t port )
```

Returns alternate modes custom SVID.

Parameters

<i>port</i>	PD port index.
-------------	----------------

Returns

SVID value.

6.2.5.13 get_mode_info()

```
alt_mode_info_t* get_mode_info (
    uint8_t port,
    uint8_t alt_mode_idx )
```

Get alt mode info.

Parameters

<i>port</i>	PD port index.
<i>alt_mode_idx</i>	alt mode index.

Returns

pointer to the alternate mode info.

6.2.5.14 get_svid_from_idx()

```
uint16_t get_svid_from_idx (
    uint8_t port,
    uint8_t idx )
```

Returns SVID which is at index position in configuration table if such SVID is available.

Parameters

<i>port</i>	PD port index.
<i>idx</i>	index of the SVID.

Returns

SVID value.

6.2.5.15 get_vdm_buff()

```
dpm_pd_cmd_buf_t* get_vdm_buff (
    uint8_t port )
```

Returns pointer to VDM buffer.

Parameters

<i>port</i>	Index of Type-C port.
-------------	-----------------------

Returns

Pointer to VDM buffer.

6.2.5.16 is_alt_mode_mgr_idle()

```
bool is_alt_mode_mgr_idle (
    uint8_t port )
```

Check whether the VDM manager for the selected port is idle.

Parameters

<i>port</i>	Port index the function is performed for.
-------------	---

Returns

true if manager is busy, false - if idle.

6.2.5.17 is_svid_supported()

```
uint8_t is_svid_supported (
    uint16_t svid,
    uint8_t port )
```

Check for presence of alt modes for given svid in alt modes compatibility table.

Parameters

<i>svid</i>	Received SVID.
<i>port</i>	Index of Type-C port.

Returns

index of supported SVID by CCG.

6.2.5.18 reg_alt_mode_mgr()

```
vdm_task_t reg_alt_mode_mgr (
    uint8_t port,
    atch_tgt_info_t * atch_tgt_info,
    vdm_msg_info_t * vdm_msg_info )
```

This function register pointers to attached dev/ama/cable info and run alt modes manager if success.

Parameters

<i>port</i>	Port index the function is performed for.
<i>atch_tgt_info</i>	Pointer to struct which holds discovery info about attached targets.
<i>vdm_msg_info</i>	Pointer to struct which holds info of received/sent VDM

Returns

VDM manager task VDM_TASK_ALT_MODE if CCG support any alternate mode. If CCG doesn't support alternate modes function returns VDM_TASK_EXIT.

6.2.5.19 reset_alt_mode_info()

```
void reset_alt_mode_info (
    alt_mode_info_t * info )
```

Resets alternate mode command info structure.

Parameters

<i>info</i>	Pointer to alternate mode info structure.
-------------	---

Returns

None.

6.2.5.20 set_alt_mode_mask()

```
void set_alt_mode_mask (
    uint8_t port,
    uint16_t mask )
```

Set mask for alternate modes which should be enabled.

Parameters

<i>port</i>	PD port index.
<i>mask</i>	mask.

Returns

void.

6.2.5.21 set_custom_svid()

```
bool set_custom_svid (
    uint8_t port,
    uint16_t svid )
```

Set alternate modes custom SVID.

Parameters

<i>port</i>	PD port index.
<i>svid</i>	SVID.

Returns

true if the SVID is set otherwise false.

6.2.5.22 vdm_task_mgr_alt_mode_process()

```
vdm_task_t vdm_task_mgr_alt_mode_process (
    uint8_t port,
    vdm_evt_t vdm_evt )
```

This function uses by DFP VDM manager to run alt modes manager processing.

Parameters

<i>port</i>	Port index the function is performed for.
<i>vdm_evt</i>	Current DFP VDM manager event.

Returns

DFP VDM manager task based on alt modes manager processing results.

6.3 app/alt_mode/custom_hpi_vid.h File Reference

```
#include <pd.h>
#include <alt_modes_mgr.h>
```

Macros

- #define [MAX_HPL_AM_VDO_NUMB](#) (1u)
- #define [HPL_AM_SVID](#) (0x0000u)

Enumerations

- enum `hpi_am_state_t` { `HPI_AM_STATE_IDLE` = 0, `HPI_AM_STATE_ENTER` = 4, `HPI_AM_STATE_EXIT` = 5, `HPI_AM_STATE_ATT` = 6 }

Functions

- `alt_mode_info_t * reg_hpi_modes` (uint8_t port, `alt_mode_reg_info_t *reg_info`)

6.3.1 Detailed Description

Custom HPI alternate mode handler header file.

6.3.2 Macro Definition Documentation

6.3.2.1 HPI_AM_SVID

```
#define HPI_AM_SVID (0x0000u)
```

SVID value assigned to HPI based alternate mode. 0 indicates that actual SVID value is taken from the configuration table or HPI register.

6.3.2.2 MAX_HPI_AM_VDO_NUMB

```
#define MAX_HPI_AM_VDO_NUMB (1u)
```

Number of Data Objects used in VDMs associated with HPI based alternate mode implementation.

6.3.3 Enumeration Type Documentation

6.3.3.1 hpi_am_state_t

```
enum hpi_am_state_t
```

This enumeration holds all possible custom HPI alt mode states.

Enumerator

<code>HPI_AM_STATE_IDLE</code>	Idle state.
<code>HPI_AM_STATE_ENTER</code>	Enter mode state.
<code>HPI_AM_STATE_EXIT</code>	Exit mode state.
<code>HPI_AM_STATE_ATT</code>	Attention state.

6.3.4 Function Documentation

6.3.4.1 reg_hpi_modes()

```
alt_mode_info_t* reg_hpi_modes (
    uint8_t port,
    alt_mode_reg_info_t * reg_info )
```

This function analyses Discovery information to find out if further custom HPI alternative mode processing is allowed.

Parameters

<i>port</i>	Port index the function is performed for.
<i>reg_info</i>	Pointer to structure which holds alt mode register info.

Returns

Pointer to custom HPI alternative mode command structure if analysis passed successful. In case of failure, function returns NULL pointer.

6.4 app/alt_mode/dp_sid.h File Reference

```
#include <pd.h>
#include <alt_modes_mngr.h>
```

Macros

- #define DP_SVID (0xFF01u)
- #define DP_ALT_MODE_ID (0u)
- #define MAX_DP_VDO_NUMB (1u)
- #define DP_VDO_IDX (0u)
- #define STATUS_UPDATE_VDO (0x01u)
- #define DFP_D_CONN (0x01u)
- #define UFP_D_CONN (0x02u)
- #define DP_1_3_SIGNALING (0x0001u)
- #define DP_CONFIG_SELECT (2u)
- #define DP_CONFIG_SELECT_DPSRC (1u)
- #define USB_CONFIG_SELECT (0u)
- #define DP_USB_SS_CONFIG (0u)
- #define DP_DFP_D_CONFIG_C (4u)
- #define DP_DFP_D_CONFIG_D (8u)
- #define DP_DFP_D_CONFIG_E (0x10u)
- #define DP_DFP_D_CONFIG_F (0x20u)
- #define DP_INVALID_CFG (0xFFu)
- #define HPD_LOW_IRQ_LOW (0x0u)
- #define HPD_HIGH_IRQ_LOW (0x1u)
- #define HPD_LOW_IRQ_HIGH (0x2u)
- #define HPD_HIGH_IRQ_HIGH (0x3u)
- #define HPD_STATE_BIT_POS (0x10u)
- #define HPD_IRQ_BIT_POS (0x11u)
- #define DP_QUEUE_STATE_SIZE (2u)
- #define DP_HPD_STATE_MASK (0x0003u)
- #define DP_QUEUE_EMPTY_INDEX (0x0u)
- #define DP_QUEUE_FULL_INDEX (7u)

- #define `DP_UFP_MAX_QUEUE_SIZE` (4u)
- #define `DP_MAX_IRQ_SIZE` (2u)
- #define `GET_HPD_IRQ_STAT(status)` (((status) >> 7u) & 0x3)
- #define `DP_SINK_CTRL_CMD` (3u)
- #define `DP_MUX_CTRL_CMD` (1u)
- #define `DP_APP_CFG_CMD` (2u)
- #define `DP_APP_VCONN_SWAP_CFG_CMD` (4u)
- #define `DP_APP_CFG_USB_IDX` (6u)
- #define `DP_APP_CFG_CMD_MAX_NUMB` (6u)
- #define `DP_ALLOWED_MUX_CONFIG_EVT` (1u)
- #define `DP_STATUS_UPDATE_EVT` (2u)
- #define `DP_CFG_CMD_ACK_MASK` (0x200)

Enumerations

- enum `dp_state_t` {
`DP_STATE_IDLE = 0, DP_STATE_ENTER = 4, DP_STATE_EXIT = 5, DP_STATE_ATT = 6,`
`DP_STATE_STATUS_UPDATE = 16, DP_STATE_CONFIG = 17 }`
- enum `dp_port_cap_t` { `DP_PORT_CAP_RSVD = 0, DP_PORT_CAP_UFP_D, DP_PORT_CAP_DFP_D,`
`DP_PORT_CAP_BOTH }`
- enum `dp_conn_t` { `DP_CONN_NONE = 0, DP_CONN_DFP_D, DP_CONN_UFP_D, DP_CONN_BOTH }`
- enum `dp_stat_bm_t` {
`DP_STAT_DFP_CONN = 0, DP_STAT_UFP_CONN, DP_STAT_PWR_LOW, DP_STAT_EN,`
`DP_STAT_MF, DP_STAT_USB_CNFG, DP_STAT_EXIT, DP_STAT_HPD,`
`DP_STAT_IRQ }`

Functions

- `alt_mode_info_t * reg_dp_modes` (uint8_t port, alt_mode_reg_info_t *reg_info)

6.4.1 Detailed Description

DisplayPort alternate mode handler header file.

6.4.2 Macro Definition Documentation

6.4.2.1 DFP_D_CONN

```
#define DFP_D_CONN (0x01u)
```

DP Status Update command VDO when DP is DFP (DFP_D DP data role).

6.4.2.2 DP_1_3_SIGNALING

```
#define DP_1_3_SIGNALING (0x0001u)
```

Standard DP signaling type value in the signaling field for Config VDO.

6.4.2.3 DP_ALLOWED_MUX_CONFIG_EVT

```
#define DP_ALLOWED_MUX_CONFIG_EVT (1u)
```

DP allowed MUX Configuration APP event value.

6.4.2.4 DP_ALT_MODE_ID

```
#define DP_ALT_MODE_ID (0u)
```

Unique ID assigned to DP alternate mode in the CCGx SDK.

6.4.2.5 DP_APP_CFG_CMD

```
#define DP_APP_CFG_CMD (2u)
```

DP MUX Configure APP command value.

6.4.2.6 DP_APP_CFG_CMD_MAX_NUMB

```
#define DP_APP_CFG_CMD_MAX_NUMB (6u)
```

Maximum vualue of USB configuration in APP DP MUX Configure command data.

6.4.2.7 DP_APP_CFG_USB_IDX

```
#define DP_APP_CFG_USB_IDX (6u)
```

Bit index of USB configuration in APP DP MUX Configure command data.

6.4.2.8 DP_APP_VCONN_SWAP_CFG_CMD

```
#define DP_APP_VCONN_SWAP_CFG_CMD (4u)
```

DP UFP Vconn Swap command value.

6.4.2.9 DP_CFG_CMD_ACK_MASK

```
#define DP_CFG_CMD_ACK_MASK (0x200)
```

Acked field mask for EC DP MUX Configure command data.

6.4.2.10 DP_CONFIG_SELECT

```
#define DP_CONFIG_SELECT (2u)
```

Value when DFP set configuration for UFP_U as UFP_D in the configuration select field for Config VDO.

6.4.2.11 DP_CONFIG_SELECT_DPSRC

```
#define DP_CONFIG_SELECT_DPSRC (1u)
```

Value when DFP set configuration for UFP_U as DFP_D in the configuration select field of Config VDO.

6.4.2.12 DP_DFP_D_CONFIG_C

```
#define DP_DFP_D_CONFIG_C (4u)
```

Pin assignment mask for C pin assignment in the Configure Pin Assignment field for Config VDO.

6.4.2.13 DP_DFP_D_CONFIG_D

```
#define DP_DFP_D_CONFIG_D (8u)
```

Pin assignment mask for D pin assignment in the Configure Pin Assignment field for Config VDO.

6.4.2.14 DP_DFP_D_CONFIG_E

```
#define DP_DFP_D_CONFIG_E (0x10u)
```

Pin assignment mask for E pin assignment in the Configure Pin Assignment field for Config VDO.

6.4.2.15 DP_DFP_D_CONFIG_F

```
#define DP_DFP_D_CONFIG_F (0x20u)
```

Pin assignment mask for F pin assignment in the Configure Pin Assignment field for Config VDO.

6.4.2.16 DP_HPD_STATE_MASK

```
#define DP_HPD_STATE_MASK (0x0003u)
```

Mask to extract the first element from the HPD queue.

6.4.2.17 DP_INVALID_CFG

```
#define DP_INVALID_CFG (0xFFu)
```

Internal DP denotation for invalid pin assignment.

6.4.2.18 DP_MAX_IRQ_SIZE

```
#define DP_MAX_IRQ_SIZE (2u)
```

Maximum number of IRQ entries in the HPD queue.

6.4.2.19 DP_MUX_CTRL_CMD

```
#define DP_MUX_CTRL_CMD (1u)
```

DP MUX Control APP command value.

6.4.2.20 DP_QUEUE_EMPTY_INDEX

```
#define DP_QUEUE_EMPTY_INDEX (0x0u)
```

Flag to indicate an empty HPD queue.

6.4.2.21 DP_QUEUE_FULL_INDEX

```
#define DP_QUEUE_FULL_INDEX (7u)
```

Flag to indicate a full HPD queue.

6.4.2.22 DP_QUEUE_STATE_SIZE

```
#define DP_QUEUE_STATE_SIZE (2u)
```

Size of one element (in bits) in the HPD queue.

6.4.2.23 DP_SINK_CTRL_CMD

```
#define DP_SINK_CTRL_CMD (3u)
```

DP Sink Control APP command value.

6.4.2.24 DP_STATUS_UPDATE_EVT

```
#define DP_STATUS_UPDATE_EVT (2u)
```

DP Status Update APP event value.

6.4.2.25 DP_SVID

```
#define DP_SVID (0xFF01u)
```

DisplayPort SVID.

6.4.2.26 DP_UFP_MAX_QUEUE_SIZE

```
#define DP_UFP_MAX_QUEUE_SIZE (4u)
```

Maximum number of entries in the HPD queue.

6.4.2.27 DP_USB_SS_CONFIG

```
#define DP_USB_SS_CONFIG (0u)
```

Pin assignment mask for USB pin assignment in the Configure Pin Assignment field for Config VDO.

6.4.2.28 DP_VDO_IDX

```
#define DP_VDO_IDX (0u)
```

Index of VDO buffer element which is used to handle DP VDO.

6.4.2.29 GET_HPDP_IRQ_STAT

```
#define GET_HPDP_IRQ_STAT(  
    status ) (((status) >> 7u) & 0x3 )
```

Macros to get HPD state from Attention/Status Update VDO.

6.4.2.30 HPD_HIGH_IRQ_HIGH

```
#define HPD_HIGH_IRQ_HIGH (0x3u)
```

HPD high and IRQ high value obtained from UFP Attention/Status Update VDO.

6.4.2.31 HPD_HIGH_IRQ_LOW

```
#define HPD_HIGH_IRQ_LOW (0x1u)
```

HPD high and IRQ low value obtained from UFP Attention/Status Update VDO.

6.4.2.32 HPD_IRQ_BIT_POS

```
#define HPD_IRQ_BIT_POS (0x11u)
```

HPD LVL Bit Position.

6.4.2.33 HPD_LOW_IRQ_HIGH

```
#define HPD_LOW_IRQ_HIGH (0x2u)
```

HPD low and IRQ high value obtained from UFP Attention/Status Update VDO.

6.4.2.34 HPD_LOW_IRQ_LOW

```
#define HPD_LOW_IRQ_LOW (0x0u)
```

HPD low and IRQ low value obtained from UFP Attention/Status Update VDO.

6.4.2.35 HPD_STATE_BIT_POS

```
#define HPD_STATE_BIT_POS (0x10u)
```

HPD LVL Bit Position.

6.4.2.36 MAX_DP_VDO_NUMB

```
#define MAX_DP_VDO_NUMB (1u)
```

Maximum number of VDOs DP uses for the alt mode flow.

6.4.2.37 STATUS_UPDATE_VDO

```
#define STATUS_UPDATE_VDO (0x01u)
```

DP Status Update command VDO used by DFP_U.

6.4.2.38 UFP_D_CONN

```
#define UFP_D_CONN (0x02u)
```

UFP_D DP data role configuration.

6.4.2.39 USB_CONFIG_SELECT

```
#define USB_CONFIG_SELECT (0u)
```

Value when DFP set configuration as USB in the configuration select field for Config VDO.

6.4.3 Enumeration Type Documentation

6.4.3.1 dp_conn_t

```
enum dp_conn_t
```

This enumeration holds possible DFP_D/UFP_D Connected status (Status Update message).

Enumerator

DP_CONN_NONE	Neither DFP_D nor UFP_D is connected.
DP_CONN_DFP_D	DFP_D is connected.
DP_CONN_UFP_D	UFP_D is connected.
DP_CONN_BOTH	Both DFP_D and UFP_D are connected.

6.4.3.2 dp_port_cap_t

```
enum dp_port_cap_t
```

This enumeration holds possible DP capabilities.

Enumerator

DP_PORT_CAP_RSVD	Reserved capability.
DP_PORT_CAP_UFP_D	UFP is UFP_D-capable.
DP_PORT_CAP_DFP_D	UFP is DFP_D-capable.
DP_PORT_CAP_BOTH	UFP is DFP_D and UFP-D capable.

6.4.3.3 dp_stat_bm_t

```
enum dp_stat_bm_t
```

This enumeration holds corresponding bit positions of Status Update VDM fields.

Enumerator

DP_STAT_DFP_CONN	DFP_D is connected field bit position.
DP_STAT_UFP_CONN	UFP_D is connected field bit position.
DP_STAT_PWR_LOW	Power Low field bit position.

Enumerator

DP_STAT_EN	Enabled field bit position.
DP_STAT_MF	Multi-function Preferred field bit position.
DP_STAT_USB_CNFG	USB Configuration Request field bit position.
DP_STAT_EXIT	Exit DP Request field bit position.
DP_STAT_HPD	HPD state field bit position.
DP_STAT_IRQ	HPD IRQ field bit position.

6.4.3.4 dp_state_t

```
enum dp_state_t
```

This enumeration holds all possible DP states.

Enumerator

DP_STATE_IDLE	Idle state.
DP_STATE_ENTER	Enter mode state.
DP_STATE_EXIT	Exit mode state.
DP_STATE_ATT	DP Attention state.
DP_STATE_STATUS_UPDATE	DP Status Update state.
DP_STATE_CONFIG	DP Configure state.

6.4.4 Function Documentation

6.4.4.1 reg_dp_modes()

```
alt_mode_info_t* reg_dp_modes (
    uint8_t port,
    alt_mode_reg_info_t * reg_info )
```

This function analyses Discovery information to find out if further DP alternative mode processing is allowed.

Parameters

<i>port</i>	Port index the function is performed for.
<i>reg_info</i>	Pointer to structure which holds alt mode register info.

Returns

Pointer to DP alternative mode command structure if analysis passed successful. In case of failure, function returns NULL pointer.

6.5 app/alt_mode/vdm_task_mgr.h File Reference

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

```
#include <pd.h>
#include <config.h>
```

Data Structures

- struct [atch_tgt_info_t](#)
- struct [vdm_msg_info_t](#)

Macros

- #define [PD_SVID_ID_HDR_VDO_START_IDX](#) (4u)
- #define [PD_DISC_ID_AMA_VDO_IDX](#) (3u)
- #define [MAX_SVID_VDO_SUPP](#) (32u)
- #define [MAX_CABLE_SVID_SUPP](#) (4u)
- #define [STD_SVID](#) (0xFF00u)
- #define [MAX_DISC_SVID_COUNT](#) (10u)
- #define [DATA_RST_RETRY_NUMB](#) (5u)
- #define [USB4_EUDO_HOST_PARAM_SHIFT](#) (13u)

Enumerations

- enum [vdm_task_t](#) {
[VDM_TASK_WAIT](#) = 0, [VDM_TASK_INIT](#), [VDM_TASK_DISC_ID](#), [VDM_TASK_DISC_SVID](#),
[VDM_TASK_REG_ATCH_TGT_INFO](#), [VDM_TASK_USB4_TBT](#), [VDM_TASK_EXIT](#), [VDM_TASK_SEND_MSG](#),
[VDM_TASK_ALT_MODE](#) }
- enum [vdm_evt_t](#) { [VDM_EVT_RUN](#) = 0, [VDM_EVT_EVAL](#), [VDM_EVT_FAIL](#), [VDM_EVT_EXIT](#) }
- enum [usb4_flag_t](#) {
[USB4_NONE](#) = 0, [USB4_FAILED](#), [USB4_PENDING](#), [USB4_TBT_CBL_FIND](#),
[USB4_TBT_CBL_DISC_RUN](#), [USB4_TBT_CBL_ENTER_SOP_P](#), [USB4_TBT_CBL_ENTER_SOP_DP](#) }

Functions

- void [enable_vdm_task_mgr](#) (uint8_t port)
- void [vdm_task_mgr](#) (uint8_t port)
- void [vdm_task_mgr_deinit](#) (uint8_t port)
- void [vdm_task_mgr_exit_modes](#) (uint8_t port)
- bool [is_vdm_task_idle](#) (uint8_t port)
- bool [is_ufp_disc_started](#) (uint8_t port)
- uint8_t * [vdm_get_disc_id_resp](#) (uint8_t port, uint8_t *resp_len_p)
- uint8_t * [vdm_get_disc_svid_resp](#) (uint8_t port, uint8_t *resp_len_p)
- void [enter_usb4](#) (uint8_t port, [sop_t](#) sop_type)
- uint32_t [usb4_update_data_status](#) (uint8_t port, [pd_do_t](#) eudo, uint32_t val)

Variables

- bool [modal_op_support](#)

6.5.1 Detailed Description

VDM task manager header file.

6.5.2 Macro Definition Documentation

6.5.2.1 DATA_RST_RETRY_NUMB

```
#define DATA_RST_RETRY_NUMB (5u)
```

Maximum number of Data Reset command retries.

6.5.2.2 MAX_CABLE_SVID_SUPP

```
#define MAX_CABLE_SVID_SUPP (4u)
```

Maximum number of cable SVIDs VDM task manager can hold in the memory.

6.5.2.3 MAX_DISC_SVID_COUNT

```
#define MAX_DISC_SVID_COUNT (10u)
```

Maximum number of DISCOVER_SVID attempts that will be performed.

6.5.2.4 MAX_SVID_VDO_SUPP

```
#define MAX_SVID_VDO_SUPP (32u)
```

Maximum number of attached target SVIDs VDM task manager can hold in the memory.

6.5.2.5 PD_DISC_ID_AMA_VDO_IDX

```
#define PD_DISC_ID_AMA_VDO_IDX (3u)
```

Index of AMA VDO in DISCOVER_ID response from port partner (after skipping the VDM header).

6.5.2.6 PD_SVID_ID_HDR_VDO_START_IDX

```
#define PD_SVID_ID_HDR_VDO_START_IDX (4u)
```

Start index of received VDM packet in case of response to DISC SVID command.

6.5.2.7 STD_SVID

```
#define STD_SVID (0xFF00u)
```

Standard vendor ID allocated to USB PD specification.

6.5.2.8 USB4_EUDO_HOST_PARAM_SHIFT

```
#define USB4_EUDO_HOST_PARAM_SHIFT (13u)
```

Bit shift to match host_supp configuration parameter with USB4 EUDO .

6.5.3 Enumeration Type Documentation

6.5.3.1 usb4_flag_t

enum `usb4_flag_t`

This enumeration lists the various vdm manager flags due to USB4 related handling.

Enumerator

USB4_NONE	Empty flag .
USB4_FAILED	This flag indicates of USB4 discovery procedure failure.
USB4_PENDING	This flag indicates that USB4 entry handling should be processed.
USB4_TBT_CBL_FIND	This flag is responsible for initiating of fibding TBT VID in cable Disc SVID response.
USB4_TBT_CBL_DISC_RUN	This flag is responsible for initiating of TBT cable Disc mode.

6.5.3.2 vdm_evt_t

enum `vdm_evt_t`

This enumeration lists the various VDM manager events to handle VDMs.

Enumerator

VDM_EVT_RUN	This event is responsible for running any of DFP VDM manager task .
VDM_EVT_EVAL	This event is responsible for eveluating VDM response .
VDM_EVT_FAIL	This event notifies task manager task if VDM rersponse fails .
VDM_EVT_EXIT	This event runs exiting from VDM task manager task .

6.5.3.3 vdm_task_t

enum `vdm_task_t`

This enumeration lists the various VDM manager tasks to handle VDMs.

Enumerator

VDM_TASK_WAIT	DFP manager wait task while waiting for VDM response.
VDM_TASK_INIT	This task is responsible for inializing of VDM manager.
VDM_TASK_DISC_ID	This task is responsible for VDM Discovery ID flow.
VDM_TASK_DISC_SVID	This task is responsible for VDM Discovery SVID flow.
VDM_TASK_REG_ATCH_TGT_INFO	This task is responsible for registering of Discovery result information in alt mode manager.
VDM_TASK_USB4_TBT	This task handles the USB4 data discovery and entry.
VDM_TASK_EXIT	This task deinits VDM task manager.
VDM_TASK_SEND_MSG	This task is responsible for forming and sending VDM message .
VDM_TASK_ALT_MODE	This task is responsible for running of alt mode manager .

6.5.4 Function Documentation

6.5.4.1 enable_vdm_task_mgr()

```
void enable_vdm_task_mgr (
    uint8_t port )
```

Enables VDM task mgr functionality.

Parameters

<i>port</i>	PD port corresponding to the VDM task manager.
-------------	--

Returns

None.

6.5.4.2 enter_usb4()

```
void enter_usb4 (
    uint8_t port,
    sop_t sop_type )
```

Initiate USB4 enter mode command to port partner or cable.

Parameters

<i>port</i>	PD port index.
<i>sop_type</i>	sop packet type.

Returns

None.

6.5.4.3 is_ufp_disc_started()

```
bool is_ufp_disc_started (
    uint8_t port )
```

Starts Discover process when CCG is UFP due to PD 3.0 spec.

Parameters

<i>port</i>	PD port corresponding to the VDM task manager.
-------------	--

Returns

true if Discover process has started, false if VDM manager id busy.

6.5.4.4 is_vdm_task_idle()

```
bool is_vdm_task_idle (
    uint8_t port )
```

Check whether the VDM task for the port is idle.

Parameters

<i>port</i>	PD port corresponding to the VDM task manager.
-------------	--

Returns

None.

6.5.4.5 usb4_update_data_status()

```
uint32_t usb4_update_data_status (
    uint8_t port,
    pd_do_t eudo,
    uint32_t val )
```

Function to update the MUX settings after USB4 mode entry.

Parameters

<i>port</i>	PD port index.
<i>eudo</i>	The Enter USB Data Object used to enter USB4.
<i>val</i>	Original value of the MUX data status register.

Returns

Updated value of the MUX data status register.

6.5.4.6 vdm_get_disc_id_resp()

```
uint8_t* vdm_get_disc_id_resp (
    uint8_t port,
    uint8_t * resp_len_p )
```

Obtain the last DISC_ID response received by the CCG device from a port partner.

Parameters

<i>port</i>	PD port index.
<i>resp_len_p</i>	Length of response in PD data objects.

Returns

Pointer to the actual response data.

6.5.4.7 vdm_get_disc_svid_resp()

```
uint8_t* vdm_get_disc_svid_resp (
    uint8_t port,
    uint8_t * resp_len_p )
```

Obtain the collective DISC_SVID response received by the CCG device from a port partner.

Parameters

<i>port</i>	PD port index.
<i>resp_len_p</i>	Length of response in PD data objects.

Returns

Pointer to the actual response data.

6.5.4.8 vdm_task_mgr()

```
void vdm_task_mgr (
    uint8_t port )
```

Main VDM task mgr function.

Parameters

<i>port</i>	PD port corresponding to the VDM task manager.
-------------	--

Returns

None.

6.5.4.9 vdm_task_mgr_deinit()

```
void vdm_task_mgr_deinit (
    uint8_t port )
```

This function deinit's VDM task manager and resets all related variables.

Parameters

<i>port</i>	PD port corresponding to the VDM task manager.
-------------	--

Returns

None.

6.5.4.10 vdm_task_mgr_exit_modes()

```
void vdm_task_mgr_exit_modes (
    uint8_t port )
```

Get the alternate mode state machine to exit any active modes. This is called in response to a VConn related fault condition.

Parameters

<i>port</i>	PD port index.
-------------	----------------

Returns

None

6.5.5 Variable Documentation**6.5.5.1 modal_op_support**

```
bool modal_op_support
```

Added check for Modal operation support bit for SOP. If no modal operation, firmware will return success to UCSI command with zero length.

6.6 app/app.h File Reference

```
#include <stdint.h>
#include <pd.h>
#include <pdss_hal.h>
#include <alt_mode_hw.h>
#include <timer_id.h>
```

Data Structures

- struct [app_status_t](#)

Macros

- #define [ADC_VBUS_MIN_OVP_LEVEL](#) (6500u)
- #define [APP_MAX_SWAP_ATTEMPT_COUNT](#) (10u)
- #define [APP_PSOURCE_CF_TIMER_PERIOD](#) (100u)
- #define [APP_PSOURCE_DIS_EXT_DIS_TIMER_PERIOD](#) (10u)
- #define [APP_PSINK_DIS_TIMER_PERIOD](#) (250u)

- #define APP_PSINK_DIS_MONITOR_TIMER_PERIOD (1u)
- #define APP_PSINK_DIS_VBUS_IN_DIS_PERIOD (20u)
- #define APP_FAULT_RECOVERY_TIMER_PERIOD (100u)
- #define APP_FAULT_RECOVERY_MAX_WAIT (500u)
- #define APP_SBU_DELAYED_CONNECT_PERIOD (25u)
- #define APP_CBL_DISC_TIMER_PERIOD (100u)
- #define APP_UFP_RECOV_VCONN_SWAP_TIMER_PERIOD (1000u)
- #define APP_VDM_BUSY_TIMER_PERIOD (50u)
- #define APP_VDM_FAIL_RETRY_PERIOD (100u)
- #define APP_CABLE_POWER_UP_DELAY (55u)
- #define APP_CABLE_VDM_START_DELAY (5u)
- #define APP_AME_TIMEOUT_TIMER_PERIOD (800u)
- #define APP_DB_SNK_FET_DIS_DELAY_TIMER_PERIOD (50u)
- #define APP_BB_ON_TIMER_PERIOD (250u)
- #define APP_INITIATE_DR_SWAP_TIMER_PERIOD (5u)
- #define APP_INITIATE_PR_SWAP_TIMER_PERIOD (500u)
- #define APP_INITIATE_SEND_IRQ_CLEAR_ACK_PERIOD (1u)
- #define RIDGE_INIT_HPD_DEQUEUE_TIMER_PERIOD (1u)
- #define UCSI_CONNECT_EVENT_PERIOD (500u)
- #define APP_BC_CDP_SM_TIMER_PERIOD (30000u)
- #define APP_BC_VDP_DM_SRC_ON_PERIOD (40u)
- #define APP_BC_VDMSRC_EN_DIS_PERIOD (20u)
- #define CCG_ACTIVITY_TIMER_PERIOD (500)
- #define OTP_DEBOUNCE_PERIOD (1)
- #define TYPE_A_CUR_SENSE_TIMER_PERIOD (30000)
- #define TYPE_A_REG_SWITCH_TIMER_PERIOD (10)
- #define TYPE_A_PWM_STEP_TIMER_PERIOD (1)
- #define PB_DEBOUNCE_PERIOD (1)
- #define APP_PB_VBATT_DEBOUNCE_IN_MS (10)
- #define THROTTLE_DEBOUNCE_PERIOD (10)
- #define THROTTLE_WAIT_FOR_PD_PERIOD (500)
- #define APP_BAD_SINK_TIMEOUT_TIMER_PERIOD (1000u)
- #define APP_PSOURCE_VBUS_SET_TIMER_PERIOD (1)
- #define APP_PSOURCE_SAFE_FET_ON_MONITOR_TIMER_PERIOD (10)
- #define APP_BC_VBUS_CYCLE_TIMER_PERIOD (200u)
- #define APP_BC_SINK_CONTACT_STABLE_TIMER_PERIOD (50u)
- #define APP_BC_DCP_DETECT_TIMER_PERIOD (1100u)
- #define APP_BC_APPLE_DETECT_TIMER_PERIOD (5u)
- #define APP_BC_DP_DM_DEBOUNCE_TIMER_PERIOD (40u)
- #define APP_BC_AFC_DETECT_TIMER_PERIOD (100u)
- #define APP_BC_GLITCH_BC_DONE_TIMER_PERIOD (1500)
- #define APP_BC_GLITCH_DM_HIGH_TIMER_PERIOD (40)
- #define APP_BC_V_NEW_REQUEST_TIMER_PERIOD (200)
- #define APP_RESET_VDM_TIMER_PERIOD (2u)
- #define APP_OT_VBE_LOW_TEMP_ADDR (0x0FFFF427)
- #define APP_OT_VBE_HIGH_TEMP_ADDR (0x0FFFF426)
- #define APP_OT_VBE_25_C_TEMP_ADDR (0x0FFFF428)
- #define APP_OT_LOW_TEMP (-40)
- #define APP_OT_HIGH_TEMP (120u)
- #define APP_OT_ROOM_TEMP (25)
- #define APP_OT_DETECTION_TIMER_PERIOD (500u)
- #define APP_VCONN_SWAP_PENDING (1u)
- #define APP_DR_SWAP_PENDING (2u)
- #define APP_PR_SWAP_PENDING (4u)
- #define APP_RETIMER_DISABLE_DELAY (200u)

- #define APP_RETIMER_ENABLE_DELAY (100u)
- #define APP_AUTO_DR_SWAP_TRY_PERIOD (10u)
- #define ICL_VSYS_STABLE_WAIT_TIME (10u)
- #define TBT_MODE_EXIT_CHECK_PERIOD (5u)
- #define APP_VCONN_RECOVERY_PERIOD (500u)

Typedefs

- typedef mux_poll_status_t(* mux_poll_fnc_cbk_t) (uint8_t port)

Enumerations

- enum app_port_fault_status_mask_t { APP_PORT_FAULT_NONE = 0x00, APP_PORT_VCONN_FAULT_ACTIVE = 0x01, APP_PORT_SINK_FAULT_ACTIVE = 0x02, APP_PORT_SRC_FAULT_ACTIVE = 0x04, APP_PORT_VBUS_DROP_WAIT_ACTIVE = 0x08, APP_PORT_V5V_SUPPLY_LOST = 0x10, APP_PORT_DISABLE_IN_PR = 0x80 }
- enum app_nb_sys_pwr_state_t { NB_SYS_PWR_STATE_S0 = 0, NB_SYS_PWR_STATE_S3, NB_SYS_PWR_STATE_S4, NB_SYS_PWR_STATE_S5 }
- enum app_cfg_sub_table_type_t { CONFIG_SUB_TABLE_OVP = 0, CONFIG_SUB_TABLE_OCP = 1, CONFIG_SUB_TABLE_SCP = 2, CONFIG_SUB_TABLE_RCP = 3, CONFIG_SUB_TABLE_OTP = 4, CONFIG_SUB_TABLE_VCONN_OCP = 5, CONFIG_SUB_TABLE_INTL_PF = 6, CONFIG_SUB_TABLE_PD_HOST = 7, CONFIG_SUB_TABLE_TBT_HOST = 8, CONFIG_SUB_TABLE_CUST_AM = 9, CONFIG_SUB_TABLE_E_ALT_MODE = 10 }
- enum sys_hw_error_type_t { SYS_HW_ERROR_NONE = 0x00, SYS_HW_ERROR_MUX_ACCESS = 0x01, SYS_HW_ERROR_REG_ACCESS = 0x02, SYS_HW_ERROR_BAD_VOLTAGE = 0x04 }
- enum app_thermistor_type_t { APP_THERMISTOR_TYPE_NTC = 0x00, APP_THERMISTOR_TYPE_PTC = 0x01, APP_THERMISTOR_TYPE_INTRNL = 0x02, APP_THERMISTOR_TYPE_ERROR = 0x03 }

Functions

- void app_init (void)
- bool app_is_port_enabled (uint8_t port)
- uint8_t app_task (uint8_t port)
- app_cbk_t * app_get_callback_ptr (uint8_t port)
- void app_event_handler (uint8_t port, app_evt_t evt, const void *dat)
- app_resp_t * app_get_resp_buf (uint8_t port)
- app_status_t * app_get_status (uint8_t port)
- bool app_sleep (void)
- void app_wakeup (void)
- bool system_sleep (void)
- bool vconn_enable (uint8_t port, uint8_t channel)
- void vconn_disable (uint8_t port, uint8_t channel)
- bool vconn_is_present (uint8_t port)
- bool vbus_is_present (uint8_t port, uint16_t volt, int8_t per)
- uint16_t vbus_get_value (uint8_t port)
- void vbus_discharge_on (uint8_t port)
- void vbus_discharge_off (uint8_t port)
- bool system_vconn_ocp_en (uint8_t port, PD_ADC_CB_T cbk)
- void system_vconn_ocp_dis (uint8_t port)
- void app_ovp_enable (uint8_t port, uint16_t volt_mV, bool pfet, PD_ADC_CB_T ovp_cb)
- void app_ovp_disable (uint8_t port, bool pfet)

- void `app_uvp_enable` (uint8_t port, uint16_t volt_mV, bool pfet, `PD_ADC_CB_T` uvp_cb)
- void `app_uvp_disable` (uint8_t port, bool pfet)
- void `app_update_bc_src_support` (uint8_t port, uint8_t enable)
- void `app_update_sys_pwr_state` (uint8_t state)
- `ccg_status_t` `app_disable_pd_port` (uint8_t port, `dpm_typec_cmd_cbk_t` cbk)
- bool `app_validate_configtable_offsets` (void)
- void `ccg_app_task_init` (void)
- void `app_bc_12_sm_start` (uint8_t port)
- void `sln_pd_event_handler` (uint8_t port, `app_evt_t` evt, const void *data)
- bool `mux_ctrl_init` (uint8_t port)
- bool `mux_ctrl_set_cfg` (uint8_t port, `mux_select_t` cfg, uint8_t polarity)
- void `mux_ctrl_bb_enable` (uint8_t port, uint8_t polarity)
- bool `fault_handler_init_vars` (uint8_t port)
- void `fault_handler_clear_counts` (uint8_t port)
- void `fault_handler_task` (uint8_t port)
- bool `fault_event_handler` (uint8_t port, `app_evt_t` evt, const void *dat)
- void `app_otp_enable` (uint8_t port)
- void `app_otp_check_temp` (uint8_t port)
- bool `app_otp_status` (uint8_t port)
- void `app_conf_for_faulty_dev_removal` (uint8_t port)
- bool `app_vdm_layer_reset` (uint8_t port)
- bool `app_port_fault_count_exceeded` (uint8_t port)
- void `vconn_change_handler` (uint8_t port, bool vconn_on)
- bool `app_is_host_hpd_virtual` (uint8_t port)
- void `app_power_share_init` (void)
- void `app_contract_handler` (uint8_t port)
- void `app_connect_change_handler` (uint8_t port)
- uint8_t * `pd_get_ptr_cfg_sub_tbl` (uint8_t port, uint8_t type)

6.6.1 Detailed Description

PD application handler header file.

6.6.2 Macro Definition Documentation

6.6.2.1 ADC_VBUS_MIN_OVP_LEVEL

```
#define ADC_VBUS_MIN_OVP_LEVEL (6500u)
```

Minimum OVP detection voltage when ADC is used to implement OVP (applies to CCG4).

6.6.2.2 APP_AME_TIMEOUT_TIMER_PERIOD

```
#define APP_AME_TIMEOUT_TIMER_PERIOD (800u)
```

tAME timer period (in ms).

6.6.2.3 APP_AUTO_DR_SWAP_TRY_PERIOD

```
#define APP_AUTO_DR_SWAP_TRY_PERIOD (10u)
```

Wait time between DR_Swap tries

6.6.2.4 APP_BAD_SINK_TIMEOUT_TIMER_PERIOD

```
#define APP_BAD_SINK_TIMEOUT_TIMER_PERIOD (1000u)
```

PD bad sink timeout timer period in ms.

6.6.2.5 APP_BB_ON_TIMER_PERIOD

```
#define APP_BB_ON_TIMER_PERIOD (250u)
```

Billboard ON delay timer period. This should be long enough for a host to properly recognize the disconnection and reconnection of the USB Billboard device.

6.6.2.6 APP_BC_AFC_DETECT_TIMER_PERIOD

```
#define APP_BC_AFC_DETECT_TIMER_PERIOD (100u)
```

AFC detection time.

6.6.2.7 APP_BC_APPLE_DETECT_TIMER_PERIOD

```
#define APP_BC_APPLE_DETECT_TIMER_PERIOD (5u)
```

Debounce time to verify if the attached device is Apple or not. Apple devices create a glitch on DP line whereas BC devices continue to drive DP lower. This period is used when Apple and BC 1.2 source protocols are supported together.

6.6.2.8 APP_BC_CDP_SM_TIMER_PERIOD

```
#define APP_BC_CDP_SM_TIMER_PERIOD (30000u)
```

CDP state machine timeout period.

6.6.2.9 APP_BC_DCP_DETECT_TIMER_PERIOD

```
#define APP_BC_DCP_DETECT_TIMER_PERIOD (1100u)
```

Tglitch_done time waiting for the portable device to complete detection. This is used by QC/AFC devices to proceed with subsequent detection.

6.6.2.10 APP_BC_DP_DM_DEBOUNCE_TIMER_PERIOD

```
#define APP_BC_DP_DM_DEBOUNCE_TIMER_PERIOD (40u)
```

Debounce time for identifying state change for DP and DM lines.

6.6.2.11 APP_BC_GLITCH_BC_DONE_TIMER_PERIOD

```
#define APP_BC_GLITCH_BC_DONE_TIMER_PERIOD (1500)
```

TGLITCH_BC_DONE timer period. This timer is used in sink mode to detect QC charger.

6.6.2.12 APP_BC_GLITCH_DM_HIGH_TIMER_PERIOD

```
#define APP_BC_GLITCH_DM_HIGH_TIMER_PERIOD (40)
```

T_GLITCH_DM_HIGH timer period. After DCP opens D+/- short, sink shall wait for this time before requesting VBUS.

6.6.2.13 APP_BC_SINK_CONTACT_STABLE_TIMER_PERIOD

```
#define APP_BC_SINK_CONTACT_STABLE_TIMER_PERIOD (50u)
```

Sink DCD stable time period.

6.6.2.14 APP_BC_V_NEW_REQUEST_TIMER_PERIOD

```
#define APP_BC_V_NEW_REQUEST_TIMER_PERIOD (200)
```

T_V_NEW_REQUEST timer period. After entering QC mode, sink must wait this much time before requesting next voltage.

6.6.2.15 APP_BC_VBUS_CYCLE_TIMER_PERIOD

```
#define APP_BC_VBUS_CYCLE_TIMER_PERIOD (200u)
```

VBUS OFF time to do a VBUS power cycle.

6.6.2.16 APP_BC_VDMSRC_EN_DIS_PERIOD

```
#define APP_BC_VDMSRC_EN_DIS_PERIOD (20u)
```

VDM_SRC enable/disable maximum period.

6.6.2.17 APP_BC_VDP_DM_SRC_ON_PERIOD

```
#define APP_BC_VDP_DM_SRC_ON_PERIOD (40u)
```

VDP_SRC or VDM_SRC minimum turn on time.

6.6.2.18 APP_CABLE_POWER_UP_DELAY

```
#define APP_CABLE_POWER_UP_DELAY (55u)
```

Time allowed for cable power up to be complete.

6.6.2.19 APP_CABLE_VDM_START_DELAY

```
#define APP_CABLE_VDM_START_DELAY (5u)
```

Cable query delay period in ms.

6.6.2.20 APP_CBL_DISC_TIMER_PERIOD

```
#define APP_CBL_DISC_TIMER_PERIOD (100u)
```

Delay to be used between cable discovery init commands.

6.6.2.21 APP_DB_SNK_FET_DIS_DELAY_TIMER_PERIOD

```
#define APP_DB_SNK_FET_DIS_DELAY_TIMER_PERIOD (50u)
```

Dead battery Sink Fet disable delay timer period.

6.6.2.22 APP_DR_SWAP_PENDING

```
#define APP_DR_SWAP_PENDING (2u)
```

Data Role swap pending bit in the app. level pending swaps flag.

6.6.2.23 APP_FAULT_RECOVERY_MAX_WAIT

```
#define APP_FAULT_RECOVERY_MAX_WAIT (500u)
```

Time for which VBus will be monitored to ensure removal of VBus by a faulty power source.

6.6.2.24 APP_FAULT_RECOVERY_TIMER_PERIOD

```
#define APP_FAULT_RECOVERY_TIMER_PERIOD (100u)
```

Period of VBus presence checks after a fault (Over-Voltage) detection while in a sink contract.

6.6.2.25 APP_INITIATE_DR_SWAP_TIMER_PERIOD

```
#define APP_INITIATE_DR_SWAP_TIMER_PERIOD (5u)
```

Time delay between successive DR_SWAP attempts (in ms).

6.6.2.26 APP_INITIATE_PR_SWAP_TIMER_PERIOD

```
#define APP_INITIATE_PR_SWAP_TIMER_PERIOD (500u)
```

Time delay between successive PR_SWAP attempts (in ms). PR_SWAP attempts are kept slow to allow other sequences such as alternate mode negotiation to complete.

6.6.2.27 APP_INITIATE_SEND_IRQ_CLEAR_ACK_PERIOD

```
#define APP_INITIATE_SEND_IRQ_CLEAR_ACK_PERIOD (1u)
```

Timer period for initiating Virtual HPD IRQ CLEAR ACK.

6.6.2.28 APP_MAX_SWAP_ATTEMPT_COUNT

```
#define APP_MAX_SWAP_ATTEMPT_COUNT (10u)
```

Number of swap attempts to be made when port partner is sending a WAIT response.

6.6.2.29 APP_OT_DETECTION_TIMER_PERIOD

```
#define APP_OT_DETECTION_TIMER_PERIOD (500u)
```

OT detection scanning period

6.6.2.30 APP_OT_HIGH_TEMP

```
#define APP_OT_HIGH_TEMP (120u)
```

High temperature which corresponds to high temperature VBJT code stored in sflash (in Celsius degrees)

6.6.2.31 APP_OT_LOW_TEMP

```
#define APP_OT_LOW_TEMP (-40)
```

Low temperature which corresponds to low temperature VBJT code stored in sflash (in Celsius degrees)

6.6.2.32 APP_OT_ROOM_TEMP

```
#define APP_OT_ROOM_TEMP (25)
```

Room temperature used for VBJT code stored in sflash (in Celsius degrees)

6.6.2.33 APP_OT_VBE_25_C_TEMP_ADDR

```
#define APP_OT_VBE_25_C_TEMP_ADDR (0x0FFFF428)
```

25C temperature VBJT code sflash address.

6.6.2.34 APP_OT_VBE_HIGH_TEMP_ADDR

```
#define APP_OT_VBE_HIGH_TEMP_ADDR (0x0FFFF426)
```

High temperature VBJT code sflash address.

6.6.2.35 APP_OT_VBE_LOW_TEMP_ADDR

```
#define APP_OT_VBE_LOW_TEMP_ADDR (0x0FFFF427)
```

Low temperature VBJT code sflash address.

6.6.2.36 APP_PB_VBATT_DEBOUNCE_IN_MS

```
#define APP_PB_VBATT_DEBOUNCE_IN_MS (10)
```

PB debounce period in ms.

6.6.2.37 APP_PR_SWAP_PENDING

```
#define APP_PR_SWAP_PENDING (4u)
```

Power Role swap pending bit in the app. level pending swaps flag.

6.6.2.38 APP_PSINK_DIS_MONITOR_TIMER_PERIOD

```
#define APP_PSINK_DIS_MONITOR_TIMER_PERIOD (1u)
```

Period of VBus voltage checks performed during power sink disable operation.

6.6.2.39 APP_PSINK_DIS_TIMER_PERIOD

```
#define APP_PSINK_DIS_TIMER_PERIOD (250u)
```

Maximum time allowed for power sink disable operation (in ms).

6.6.2.40 APP_PSINK_DIS_VBUS_IN_DIS_PERIOD

```
#define APP_PSINK_DIS_VBUS_IN_DIS_PERIOD (20u)
```

Duration of discharge sequence on the VBUS_IN supply in CCG3PA/CCG3PA2 designs.

6.6.2.41 APP_PSOURCE_CF_TIMER_PERIOD

```
#define APP_PSOURCE_CF_TIMER_PERIOD (100u)
```

Duration of Power Source Current Foldback timer (in ms).

6.6.2.42 APP_PSOURCE_DIS_EXT_DIS_TIMER_PERIOD

```
#define APP_PSOURCE_DIS_EXT_DIS_TIMER_PERIOD (10u)
```

Duration of extra VBus discharge after voltage drops below desired level (in ms).

6.6.2.43 APP_PSOURCE_SAFE_FET_ON_MONITOR_TIMER_PERIOD

```
#define APP_PSOURCE_SAFE_FET_ON_MONITOR_TIMER_PERIOD (10)
```

Psource safe FET ON monitor timer period in ms.

6.6.2.44 APP_PSOURCE_VBUS_SET_TIMER_PERIOD

```
#define APP_PSOURCE_VBUS_SET_TIMER_PERIOD (1)
```

Power source VBUS set timer period in ms.

6.6.2.45 APP_RESET_VDM_TIMER_PERIOD

```
#define APP_RESET_VDM_TIMER_PERIOD (2u)
```

Reset VDM layer time period.

6.6.2.46 APP_RETIMER_DISABLE_DELAY

```
#define APP_RETIMER_DISABLE_DELAY (200u)
```

Time to wait after disconnect to power the retimer off

6.6.2.47 APP_RETIMER_ENABLE_DELAY

```
#define APP_RETIMER_ENABLE_DELAY (100u)
```

Time to wait after hard reset to power the retimer ON

6.6.2.48 APP_SBU_DELAYED_CONNECT_PERIOD

```
#define APP_SBU_DELAYED_CONNECT_PERIOD (25u)
```

Delay (in ms) between Thunderbolt mode entry and SBU path configuration.

6.6.2.49 APP_UFP_RECOV_VCONN_SWAP_TIMER_PERIOD

```
#define APP_UFP_RECOV_VCONN_SWAP_TIMER_PERIOD (1000u)
```

Timer period used to run Vconn swap after V5V was lost and recovered while UFP.

6.6.2.50 APP_VCONN_RECOVERY_PERIOD

```
#define APP_VCONN_RECOVERY_PERIOD (500u)
```

Delay between VConn fault and recovery attempt.

6.6.2.51 APP_VCONN_SWAP_PENDING

```
#define APP_VCONN_SWAP_PENDING (1u)
```

VConn swap pending bit in the app. level pending swaps flag.

6.6.2.52 APP_VDM_BUSY_TIMER_PERIOD

```
#define APP_VDM_BUSY_TIMER_PERIOD (50u)
```

VDM busy timer period (in ms).

6.6.2.53 APP_VDM_FAIL_RETRY_PERIOD

```
#define APP_VDM_FAIL_RETRY_PERIOD (100u)
```

VDM retry (on failure) timer period in ms.

6.6.2.54 CCG_ACTIVITY_TIMER_PERIOD

```
#define CCG_ACTIVITY_TIMER_PERIOD (500)
```

CCG activity timer period in ms.

6.6.2.55 ICL_VSYS_STABLE_WAIT_TIME

```
#define ICL_VSYS_STABLE_WAIT_TIME (10u)
```

How long to wait for VSYS to stabilize

6.6.2.56 OTP_DEBOUNCE_PERIOD

```
#define OTP_DEBOUNCE_PERIOD (1)
```

OTP debounce timer period in ms.

6.6.2.57 PB_DEBOUNCE_PERIOD

```
#define PB_DEBOUNCE_PERIOD (1)
```

PB debounce timer period in ms.

6.6.2.58 RIDGE_INIT_HPD_DEQUEUE_TIMER_PERIOD

```
#define RIDGE_INIT_HPD_DEQUEUE_TIMER_PERIOD (1u)
```

Timer period in ms for initiating virtual HPD dequeue.

6.6.2.59 TBT_MODE_EXIT_CHECK_PERIOD

```
#define TBT_MODE_EXIT_CHECK_PERIOD (5u)
```

Periodicity of checking if TBT mode should be exited

6.6.2.60 THROTTLE_DEBOUNCE_PERIOD

```
#define THROTTLE_DEBOUNCE_PERIOD (10)
```

THROTTLE time period in ms.

6.6.2.61 THROTTLE_WAIT_FOR_PD_PERIOD

```
#define THROTTLE_WAIT_FOR_PD_PERIOD (500)
```

THROTTLE time period in ms.

6.6.2.62 TYPE_A_CUR_SENSE_TIMER_PERIOD

```
#define TYPE_A_CUR_SENSE_TIMER_PERIOD (30000)
```

TYPE-A current sense timer period.

6.6.2.63 TYPE_A_PWM_STEP_TIMER_PERIOD

```
#define TYPE_A_PWM_STEP_TIMER_PERIOD (1)
```

TYPE-A port PWM step change time in ms.

6.6.2.64 TYPE_A_REG_SWITCH_TIMER_PERIOD

```
#define TYPE_A_REG_SWITCH_TIMER_PERIOD (10)
```

TYPE-A regulator switch timer period in ms.

6.6.2.65 UCSI_CONNECT_EVENT_PERIOD

```
#define UCSI_CONNECT_EVENT_PERIOD (500u)
```

Delay (in ms) between Type-C connection and sending UCSI event notification.

6.6.3 Typedef Documentation

6.6.3.1 mux_poll_fnc_cbk_t

```
typedef mux_poll_status_t(* mux_poll_fnc_cbk_t) (uint8_t port)
```

This type of the function is used by app layer to call MUX polling function.

Parameters

<i>port</i>	Port index the function is performed for.
-------------	---

Returns

MUX polling status

6.6.4 Enumeration Type Documentation

6.6.4.1 app_nb_sys_pwr_state_t

```
enum app_nb_sys_pwr_state_t
```

List of system power states for Notebook/Desktop designs. These states will be reported by the EC to CCG device through HPI registers.

Enumerator

NB_SYS_PWR_STATE_S0	Notebook/Desktop is in active (S0) state.
NB_SYS_PWR_STATE_S3	Notebook/Desktop is in Sleep (S3) state.
NB_SYS_PWR_STATE_S4	Notebook/Desktop is in Hibernate (S4) state.
NB_SYS_PWR_STATE_S5	Notebook/Desktop is in Off (S5) state.

6.6.4.2 app_port_fault_status_mask_t

```
enum app_port_fault_status_mask_t
```

Fault detection and handling related status bits tracked in the fault_status field.

Enumerator

APP_PORT_FAULT_NONE	System functioning without any fault.
APP_PORT_VCONN_FAULT_ACTIVE	Status bit that indicates VConn fault is active.
APP_PORT_SINK_FAULT_ACTIVE	Status bit that indicates sink fault handling is pending.
APP_PORT_SRC_FAULT_ACTIVE	Status bit that indicates source fault handling is pending.
APP_PORT_VBUS_DROP_WAIT_ACTIVE	Status bit that indicates wait for VBus drop is pending.
APP_PORT_V5V_SUPPLY_LOST	Status bit that indicates that V5V supply (for VConn) has been lost.
APP_PORT_DISABLE_IN_PROGRESS	Port disable operation is in progress.

6.6.4.3 app_thermistor_type_t

enum `app_thermistor_type_t`

List of possible Thermistor types that can be configured.

Enumerator

APP_THERMISTOR_TYPE_NTC	NTC Thermistor type configured.
APP_THERMISTOR_TYPE_PTC	PTC Thermistor type configured.
APP_THERMISTOR_TYPE_INTRNL	CCGx internal VBJT based temperature sensing.
APP_THERMISTOR_TYPE_ERROR	Invalid Thermistor type configured.

6.6.4.4 sys_hw_error_type_t

enum `sys_hw_error_type_t`

List of possible hardware errors defined for the system.

Enumerator

SYS_HW_ERROR_NONE	No error.
SYS_HW_ERROR_MUX_ACCESS	Error while accessing data MUX.
SYS_HW_ERROR_REG_ACCESS	Error while accessing regulator.
SYS_HW_ERROR_BAD_VOLTAGE	Unexpected voltage generated by source regulator.

6.6.5 Function Documentation

6.6.5.1 app_bc_12_sm_start()

```
void app_bc_12_sm_start (
    uint8_t port )
```

Start the simplified BC 1.2 (DCP/CDP) state machine for CCG5 device on detection of a Type-C sink connection.

Parameters

<i>port</i>	Type-C port index.
-------------	--------------------

Returns

None

6.6.5.2 app_conf_for_faulty_dev_removal()

```
void app_conf_for_faulty_dev_removal (
    uint8_t port )
```

Prepare the CCG to wait for physical detach of faulty port partner.

Parameters

<i>port</i>	PD port index.
-------------	----------------

Returns

None

6.6.5.3 app_connect_change_handler()

```
void app_connect_change_handler (
    uint8_t port )
```

This function is called whenever there is a change in the connection.

Parameters

<i>port</i>	PD port index.
-------------	----------------

Returns

void.

6.6.5.4 app_contract_handler()

```
void app_contract_handler (
    uint8_t port )
```

This function is called at the end of a PD contract to check whether any role swaps need to be triggered.

Parameters

<i>port</i>	PD port index.
-------------	----------------

Returns

void.

6.6.5.5 app_disable_pd_port()

```
ccg_status_t app_disable_pd_port (
    uint8_t port,
    dpm_typepec_cmd_cbk_t cbk )
```

Wrapper function for PD port disable. This function is used to ensure that any application level state associated with a faulty connection are cleared when the user wants to disable a PD port.

Parameters

<i>port</i>	Index of port to be disabled.
<i>cbk</i>	Callback to be called after operation is complete.

Returns

CCG_STAT_SUCCESS on success, appropriate error code otherwise.

6.6.5.6 app_event_handler()

```
void app_event_handler (
    uint8_t port,
    app_evt_t evt,
    const void * dat )
```

Handler for event notifications from the PD stack.

Parameters

<i>port</i>	Port on which events are to be handled.
<i>evt</i>	Type of event to be handled.
<i>dat</i>	Data associated with the event.

Returns

None

6.6.5.7 app_get_callback_ptr()

```
app_cbk_t* app_get_callback_ptr (
    uint8_t port )
```

This function return the App callback structure pointer.

Parameters

<i>port</i>	port index
-------------	------------

Returns

Application callback structure pointer

6.6.5.8 app_get_resp_buf()

```
app_resp_t* app_get_resp_buf (
    uint8_t port )
```


Get a handle to the application provide PD command response buffer.

Parameters

<i>port</i>	PD port corresponding to the command and response.
-------------	--

Returns

Pointer to the response buffer.

6.6.5.9 app_get_status()

```
app_status_t* app_get_status (
    uint8_t port )
```

Get handle to structure containing information about the system status for a PD port.

Parameters

<i>port</i>	PD port to be queried.
-------------	------------------------

Returns

Pointer to the system information structure.

6.6.5.10 app_init()

```
void app_init (
    void )
```

Application level init function.

This function performs any Application level initialization required for the CCG solution. This should be called before calling the dpm_init function.

Returns

None.

6.6.5.11 app_is_host_hpd_virtual()

```
bool app_is_host_hpd_virtual (
    uint8_t port )
```

Check whether the solution is configured to support virtual HPD.

Parameters

<i>port</i>	PD port index
-------------	---------------

Returns

true if virtual HPD is enabled.

6.6.5.12 app_is_port_enabled()

```
bool app_is_port_enabled (
    uint8_t port )
```

Check whether the specified PD port is enabled in the system configuration.

Parameters

<i>port</i>	PD port index.
-------------	----------------

Returns

true if the port is enabled, false otherwise.

6.6.5.13 app_otp_check_temp()

```
void app_otp_check_temp (
    uint8_t port )
```

Perform OTP check. This function is called from app task.

Parameters

<i>port</i>	PD port index.
-------------	----------------

Returns

None.

6.6.5.14 app_otp_enable()

```
void app_otp_enable (
    uint8_t port )
```

Enable the Over-Temperature Protection logic.

Parameters

<i>port</i>	PD port index.
-------------	----------------

Returns

None.

6.6.5.15 app_otp_status()

```
bool app_otp_status (
    uint8_t port )
```

Returns the status of Over-Temperature Protection.

Parameters

<i>port</i>	PD port index.
-------------	----------------

Returns

Returns True if Over-Temperature condition is active, otherwise false.

6.6.5.16 app_ovp_disable()

```
void app_ovp_disable (
    uint8_t port,
    bool pfet )
```

Disable the Over-Voltage protection circuitry.

Parameters

<i>port</i>	PD port to be configured.
<i>pfet</i>	Whether PFET is used for the power supply control.

Returns

None

6.6.5.17 app_ovp_enable()

```
void app_ovp_enable (
    uint8_t port,
    uint16_t volt_mV,
    bool pfet,
    PD_ADC_CB_T ovp_cb )
```

Enable and configure the Over-Voltage protection circuitry.

Parameters

<i>port</i>	PD port to be configured.
<i>volt_mV</i>	Expected VBus voltage.
<i>pfet</i>	Whether PFET is used for the power supply control.
<i>ovp_cb</i>	Callback function to be triggered when there is an OV event.

Returns

None

6.6.5.18 app_port_fault_count_exceeded()

```
bool app_port_fault_count_exceeded (
    uint8_t port )
```

Check whether any fault count has exceeded limit for the specified PD port.

Parameters

<i>port</i>	PD port index.
-------------	----------------

Returns

true if fault count has exceeded limit, false otherwise.

6.6.5.19 app_power_share_init()

```
void app_power_share_init (
    void )
```

Function to initialize the power sharing across ports on a dual-port CCG device.

Returns

None

6.6.5.20 app_sleep()

```
bool app_sleep (
    void )
```

Check whether the APP handlers are ready to allow device deep sleep.

Returns

true if APP handler is idle, false otherwise.

6.6.5.21 app_task()

```
uint8_t app_task (
    uint8_t port )
```

Handler for application level asynchronous tasks.

Parameters

<i>port</i>	USB-PD port for which tasks are to be handled.
-------------	--

Returns

1 in case of success, 0 in case of task handling error.

6.6.5.22 app_update_bc_src_support()

```
void app_update_bc_src_support (
    uint8_t port,
    uint8_t enable )
```

Function to update the BC 1.2 source support.

Parameters

<i>port</i>	PD port to be configured.
<i>enable</i>	Whether BC 1.2 support should be enabled or disabled.

Returns

None

6.6.5.23 app_update_sys_pwr_state()

```
void app_update_sys_pwr_state (
    uint8_t state )
```

Function to update the system power state.

Parameters

<i>state</i>	Current system power state: 0=S0, Non-zero=other states.
--------------	--

6.6.5.24 app_uvp_disable()

```
void app_uvp_disable (
    uint8_t port,
    bool pfet )
```

Disable the Under-Voltage protection circuitry.

Parameters

<i>port</i>	PD port to be configured.
<i>pfet</i>	Whether PFET is used for the power supply control.

Returns

None

6.6.5.25 app_uvp_enable()

```
void app_uvp_enable (
    uint8_t port,
    uint16_t volt_mV,
    bool pfet,
    PD_ADC_CB_T uvp_cb )
```

Enable and configure the Under-Voltage protection circuitry.

Parameters

<i>port</i>	PD port to be configured.
<i>volt_mV</i>	Expected VBus voltage.
<i>pfet</i>	Whether PFET is used for the power supply control.
<i>uvp_cb</i>	Callback function to be triggered when there is an UV event.

Returns

None

6.6.5.26 app_validate_configtable_offsets()

```
bool app_validate_configtable_offsets (
    void )
```

Validate the configuration table specified.

Each copy of CCGx firmware on the device flash contains an embedded configuration table that defines the runtime behaviour of the CCGx device. This function checks whether the configuration table located at the specified location is valid (has valid offsets).

Returns

true if the table is valid, false otherwise.

6.6.5.27 app_vdm_layer_reset()

```
bool app_vdm_layer_reset (
    uint8_t port )
```

Restarts alternate mode layer.

Parameters

<i>port</i>	PD port index.
-------------	----------------

Returns

True if the alternate mode resetted, false otherwise.

6.6.5.28 app_wakeup()

```
void app_wakeup (
    void )
```

Restore the APP handler state after CCG device wakes from deep-sleep.

Returns

None

6.6.5.29 ccg_app_task_init()

```
void ccg_app_task_init (
    void )
```

Initialize CCGx periodic application level tasks.

Returns

None

6.6.5.30 fault_event_handler()

```
bool fault_event_handler (
    uint8_t port,
    app_evt_t evt,
    const void * dat )
```

Handle any application events associated with fault handling logic.

Parameters

<i>port</i>	PD port index on which the event is received.
<i>evt</i>	Event code.
<i>dat</i>	Data associated with the event code.

Returns

true if the event does not need to be passed up to the solution layer.

6.6.5.31 fault_handler_clear_counts()

```
void fault_handler_clear_counts (
    uint8_t port )
```


Clear the fault occurrence counts after a Type-C detach is detected.

Parameters

<i>port</i>	PD port index.
-------------	----------------

Returns

None.

6.6.5.32 `fault_handler_init_vars()`

```
bool fault_handler_init_vars (  
    uint8_t port )
```

Initialize fault-handling related variables from the configuration table.

Parameters

<i>port</i>	PD port index.
-------------	----------------

Returns

false if the configuration table is invalid, true otherwise.

6.6.5.33 `fault_handler_task()`

```
void fault_handler_task (  
    uint8_t port )
```

Perform any fault handler related tasks.

Parameters

<i>port</i>	PD port on which the tasks are to be performed.
-------------	---

Returns

None

6.6.5.34 `mux_ctrl_bb_enable()`

```
void mux_ctrl_bb_enable (  
    uint8_t port,  
    uint8_t polarity )
```

Enable BB device enumeration on the board.

Parameters

<i>port</i>	PD port index.
<i>polarity</i>	PD connection (plug) orientation.

6.6.5.35 `mux_ctrl_init()`

```
bool mux_ctrl_init (
    uint8_t port )
```

Initialize the Type-C Data Mux for a specific PD port.

Parameters

<i>port</i>	USB-PD port for which the MUX is to be initialized.
-------------	---

Returns

Returns true if the MUX is initialized successfully, false otherwise.

6.6.5.36 `mux_ctrl_set_cfg()`

```
bool mux_ctrl_set_cfg (
    uint8_t port,
    mux_select_t cfg,
    uint8_t polarity )
```

Set the Type-C MUX to the desired configuration.

Parameters

<i>port</i>	PD port on which MUX is to be configured.
<i>cfg</i>	Desired MUX configuration.
<i>polarity</i>	Polarity of the Type-C connection.

Returns

Returns true if the operation is successful, false otherwise.

6.6.5.37 `pd_get_ptr_cfg_sub_tbl()`

```
uint8_t* pd_get_ptr_cfg_sub_tbl (
    uint8_t port,
    uint8_t type )
```

Function to retrieve desired sub-table from the configuration table.

Parameters

<i>port</i>	PD port index.
<i>type</i>	Type of sub-table to be retrieved.

6.6.5.38 sln_pd_event_handler()

```
void sln_pd_event_handler (
    uint8_t port,
    app_evt_t evt,
    const void * data )
```

Solution handler for PD events reported from the stack.

The function provides all PD events to the solution. For a solution supporting HPI, the solution function should re-direct the calls to hpi_pd_event_handler.

Parameters

<i>port</i>	PD port corresponding to the event.
<i>evt</i>	Event that is being notified.
<i>data</i>	Data associated with the event. This is an opaque pointer that needs to be de-referenced based on event type.

Returns

None

6.6.5.39 system_sleep()

```
bool system_sleep (
    void )
```

Function to place CCG device in power saving mode if possible.

This function places the CCG device in power saving deep sleep mode if possible. The function checks for each interface (PD, HPI etc.) being idle and then enters sleep mode with the appropriate wake-up triggers. If the device enters sleep mode, the function will only return after the device has woken up.

Returns

true if the device went into sleep, false otherwise.

6.6.5.40 system_vconn_ocp_dis()

```
void system_vconn_ocp_dis (
    uint8_t port )
```

This function disable vconn ocp.

Parameters

<i>port</i>	Port index
-------------	------------

Returns

None

6.6.5.41 system_vconn_ocp_en()

```
bool system_vconn_ocp_en (
    uint8_t port,
    PD_ADC_CB_T cbk )
```

This function enable vconn ocp.

Parameters

<i>port</i>	Port index
<i>cbk</i>	OCP callback

Returns

Returns true on success, false if parameters are invalid.

6.6.5.42 vbus_discharge_off()

```
void vbus_discharge_off (
    uint8_t port )
```

This function turns off discharge FET on selected port.

Parameters

<i>port</i>	Port index the function is performed for.
-------------	---

Returns

None

6.6.5.43 vbus_discharge_on()

```
void vbus_discharge_on (
    uint8_t port )
```

This function turns on discharge FET on selected port.

Parameters

<i>port</i>	Port index the function is performed for.
-------------	---

Returns

None

6.6.5.44 vbus_get_value()

```
uint16_t vbus_get_value (
    uint8_t port )
```

This function return current VBUS voltage in mV.

Parameters

<i>port</i>	Port index the function is performed for.
-------------	---

Returns

VBUS voltage in mV

6.6.5.45 vbus_is_present()

```
bool vbus_is_present (
    uint8_t port,
    uint16_t volt,
    int8_t per )
```

This function checks if power is present on VBus.

Parameters

<i>port</i>	Port index the function is performed for.
<i>volt</i>	Voltage in mV units.
<i>per</i>	Threshold margin.

Returns

true if power is present on VBus, else returns false

6.6.5.46 vconn_change_handler()

```
void vconn_change_handler (
    uint8_t port,
    bool vconn_on )
```

Function to handle VConn supply state change due to OCP condition or V5V supply change.

Parameters

<i>port</i>	PD port index.
<i>vconn_on</i>	Whether VConn has just turned ON or OFF.

Returns

None

6.6.5.47 vconn_disable()

```
void vconn_disable (
    uint8_t port,
    uint8_t channel )
```

This function disables VCONN power.

Parameters

<i>port</i>	Port index the function is performed for.
<i>channel</i>	Selected CC line.

Returns

None

6.6.5.48 vconn_enable()

```
bool vconn_enable (
    uint8_t port,
    uint8_t channel )
```

This function enables VCONN power.

Parameters

<i>port</i>	Port index the function is performed for.
<i>channel</i>	Selected CC line.

Returns

True if VConn was turned ON; false if NOT.

6.6.5.49 vconn_is_present()

```
bool vconn_is_present (
    uint8_t port )
```

This function checks if power is present on VConn.

Parameters

<i>port</i>	Port index the function is performed for.
-------------	---

Returns

true if power is present on VConn, else returns false

6.7 app/battery_charging.h File Reference

```
#include <config.h>
#include <pd.h>
#include <status.h>
#include <chgb_hal.h>
```

Data Structures

- union [bc_dp_dm_state_t](#)
- struct [bc_status_t](#)

Macros

- #define [BC_CMP_0_IDX](#) (0u)
- #define [BC_CMP_1_IDX](#) (1u)
- #define [BC_AMP_LIMIT](#) (300)
- #define [APPLE_AMP_1A](#) (100)
- #define [APPLE_AMP_2_1A](#) (210)
- #define [APPLE_AMP_2_4A](#) (240)
- #define [QC_AMP_5V](#) (300)
- #define [QC_AMP_9V](#) (300)
- #define [QC_AMP_12V](#) (300)
- #define [QC_AMP_20V](#) (300)
- #define [QC_AMP_CONT](#) (300)
- #define [QC_CONT_VOLT_CHANGE_PER_PULSE](#) (200u)
- #define [QC3_MIN_VOLT](#) (3400u)
- #define [CCG_POWER_PRECISION_MULT](#) (100000u)

Enumerations

- enum [bc_port_type_t](#) { [BC_PORT_TYPE_A](#) = 0, [BC_PORT_TYPE_C](#) }
- enum [bc_port_role_t](#) { [BC_PORT_SINK](#) = 0, [BC_PORT_SOURCE](#) }
- enum [bc_qc_class_t](#) { [BC_QC_CLASS_A](#) = 0, [BC_QC_CLASS_B](#) }
- enum [bc_qc_ver_t](#) { [BC_QC_VER_2](#) = 0, [BC_QC_VER_3](#) }
- enum [bc_apple_id_t](#) { [BC_APPLE_ID_1A](#) = 0, [BC_APPLE_ID_2_1A](#), [BC_APPLE_ID_2_4A](#) }
- enum [bc_charge_mode_t](#) { [BC_CHARGE_NONE](#) = 0, [BC_CHARGE_DCP](#), [BC_CHARGE_QC2](#), [BC_CHARGE_QC3](#), [BC_CHARGE_AFC](#), [BC_CHARGE_APPLE](#), [BC_CHARGE_CDP](#) }
- enum [bc_d_status_t](#) { [BC_D_GND](#) = 0, [BC_D_0_6V](#), [BC_D_3_3V](#), [BC_D_ERR](#) }
- enum [bc_apple_term](#) { [APPLE_TERM1](#) = 1, [APPLE_TERM2](#) = 2, [APPLE_TERM3](#) = 3 }

- enum `bc_apple_brick_id` {
`APPLE_BRICK_ID_0 = 0x11, APPLE_BRICK_ID_1 = 0x12, APPLE_BRICK_ID_2 = 0x13, APPLE_BRICK_ID_3 = 0x21,`
`APPLE_BRICK_ID_4 = 0x22, APPLE_BRICK_ID_5 = 0x23, APPLE_BRICK_ID_6 = 0x31, APPLE_BRICK_ID_7 = 0x32,`
`APPLE_BRICK_ID_8 = 0x33 }`
- enum `bc_state_t` {
`BC_FSM_OFF = 0, BC_FSM_SRC_LOOK_FOR_CONNECT, BC_FSM_SRC_INITIAL_CONNECT,`
`BC_FSM_SRC_APPLE_CONNECTED,`
`BC_FSM_SRC_CDP_CONNECTED, BC_FSM_SRC_OTHERS_CONNECTED, BC_FSM_SRC_QC_OR_AFC,`
`BC_FSM_SRC_QC_CONNECTED,`
`BC_FSM_SRC_AFC_CONNECTED, BC_FSM_SINK_START, BC_FSM_SINK_APPLE_CHARGER_DETECT,`
`BC_FSM_SINK_APPLE_BRICK_ID_DETECT,`
`BC_FSM_SINK_PRIMARY_CHARGER_DETECT, BC_FSM_SINK_TYPE_C_ONLY_SOURCE_CONNECTED,`
`BC_FSM_SINK_SECONDARY_CHARGER_DETECT, BC_FSM_SINK_DCP_CONNECTED,`
`BC_FSM_SINK_SDP_CONNECTED, BC_FSM_SINK_CDP_CONNECTED, BC_FSM_MAX_STATES }`
- enum `bc_sink_timer_t` { `BC_SINK_TIMER_NONE = 0, BC_SINK_DCD_DEBOUNCE_TIMER = 1 }`

Functions

- `ccg_status_t bc_init` (`uint8_t cport`)
- `const chg_cfg_params_t * bc_get_config` (`uint8_t cport`)
- `ccg_status_t bc_start` (`uint8_t cport, bc_port_role_t port_role`)
- `ccg_status_t bc_stop` (`uint8_t cport`)
- `bool bc_is_active` (`uint8_t cport`)
- `ccg_status_t bc_fsm` (`uint8_t cport`)
- `bool bc_port_is_cdp` (`uint8_t cport`)
- `bool bc_sleep` (`void`)
- `bool bc_wakeup` (`void`)
- `const bc_status_t * bc_get_status` (`uint8_t cport`)
- `void bc_pd_event_handler` (`uint8_t port, app_evt_t evt`)
- `void bc_set_bc_evt` (`uint8_t cport, uint32_t evt_mask`)
- `void bc_clear_bc_evt` (`uint8_t cport, uint32_t evt_mask`)
- `uint8_t ccg_get_system_max_pdp` (`uint8_t cport`)
- `void qc_set_cf_limit` (`uint8_t cport`)
- `void bc_afc_form_vi` (`uint8_t cport`)

6.7.1 Detailed Description

Battery Charging header file.

6.7.2 Macro Definition Documentation

6.7.2.1 APPLE_AMP_1A

```
#define APPLE_AMP_1A (100)
```

Current limit for Apple 1.0A brick.

6.7.2.2 APPLE_AMP_2_1A

```
#define APPLE_AMP_2_1A (210)
```

Current limit for Apple 2.1A brick.

6.7.2.3 APPLE_AMP_2_4A

```
#define APPLE_AMP_2_4A (240)
```

Current limit for Apple 2.4A brick.

6.7.2.4 BC_AMP_LIMIT

```
#define BC_AMP_LIMIT (300)
```

Maximum current across various BC modes: 3.0 A.

6.7.2.5 BC_CMP_0_IDX

```
#define BC_CMP_0_IDX (0u)
```

Copyright

Copyright (2014-2020), Cypress Semiconductor Corporation or a subsidiary of Cypress Semiconductor Corporation. All rights reserved.

This software, including source code, documentation and related materials (“Software”), is owned by Cypress Semiconductor Corporation or one of its subsidiaries (“Cypress”) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Therefore, you may use this Software only as provided in the license agreement accompanying the software package from which you obtained this Software (“EULA”).

If no EULA applies, Cypress hereby grants you a personal, nonexclusive, non-transferable license to copy, modify, and compile the Software source code solely for use in connection with Cypress’s integrated circuit products. Any reproduction, modification, translation, compilation, or representation of this Software except as specified above is prohibited without the express written permission of Cypress. Disclaimer: THIS SOFTWARE IS PROVIDED AS-IS, WITH NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, NONINFRINGEMENT, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes to the Software without notice. Cypress does not assume any liability arising out of the application or use of the Software or any product or circuit described in the Software. Cypress does not authorize its products for use in any products where a malfunction or failure of the Cypress product may reasonably be expected to result in significant property damage, injury or death (“High Risk Product”). By including Cypress’s product in a High Risk Product, the manufacturer of such system or application assumes all risk of such use and in doing so agrees to indemnify Cypress against all liability. Battery charger comparator #1.

6.7.2.6 BC_CMP_1_IDX

```
#define BC_CMP_1_IDX (1u)
```

Battery charger comparator #2.

6.7.2.7 CCG_POWER_PRECISION_MULT

```
#define CCG_POWER_PRECISION_MULT (100000u)
```

Multiplier used in PDP value simplification

6.7.2.8 QC3_MIN_VOLT

```
#define QC3_MIN_VOLT (3400u)
```

Minimum supply voltage used in QC charging.

6.7.2.9 QC_AMP_12V

```
#define QC_AMP_12V (300)
```

Current limit for Quick Charge at 12 V.

6.7.2.10 QC_AMP_20V

```
#define QC_AMP_20V (300)
```

Current limit for Quick Charge at 20 V.

6.7.2.11 QC_AMP_5V

```
#define QC_AMP_5V (300)
```

Current limit for Quick Charge at 5 V.

6.7.2.12 QC_AMP_9V

```
#define QC_AMP_9V (300)
```

Current limit for Quick Charge at 9 V.

6.7.2.13 QC_AMP_CONT

```
#define QC_AMP_CONT (300)
```

Current limit for Quick Charge continuous mode.

6.7.2.14 QC_CONT_VOLT_CHANGE_PER_PULSE

```
#define QC_CONT_VOLT_CHANGE_PER_PULSE (200u)
```

Quick Charge continuous mode voltage change per pulse received.

6.7.3 Enumeration Type Documentation**6.7.3.1 bc_apple_brick_id**

```
enum bc_apple_brick_id
```

List of possible Apple Brick IDs based on the terminations on DP and DM pins.

Enumerator

APPLE_BRICK_ID↔ _0	APPLE_TERM1 on DM, APPLE_TERM1 on DP.
-----------------------	---------------------------------------

Enumerator

APPLE_BRICK_ID↔ _1	APPLE_TERM1 on DM, APPLE_TERM2 on DP.
APPLE_BRICK_ID↔ _2	APPLE_TERM1 on DM, APPLE_TERM3 on DP.
APPLE_BRICK_ID↔ _3	APPLE_TERM2 on DM, APPLE_TERM1 on DP.
APPLE_BRICK_ID↔ _4	APPLE_TERM2 on DM, APPLE_TERM2 on DP.
APPLE_BRICK_ID↔ _5	APPLE_TERM2 on DM, APPLE_TERM3 on DP.
APPLE_BRICK_ID↔ _6	APPLE_TERM3 on DM, APPLE_TERM1 on DP.
APPLE_BRICK_ID↔ _7	APPLE_TERM3 on DM, APPLE_TERM2 on DP.
APPLE_BRICK_ID↔ _8	APPLE_TERM3 on DM, APPLE_TERM3 on DP.

6.7.3.2 bc_apple_id_t

```
enum bc_apple_id_t
```

Apple charger brick id.

Enumerator

BC_APPLE_ID_1A	Apple 1.0 A charging brick.
BC_APPLE_ID_2_1A	Apple 2.1 A charging brick.
BC_APPLE_ID_2_4A	Apple 2.4 A charging brick.

6.7.3.3 bc_apple_term

```
enum bc_apple_term
```

Enumeration of Apple terminations codes.

Enumerator

APPLE_TERM1	Termination 1 code: 1 - 2.22 V
APPLE_TERM2	Termination 2 code: 2.22 - 2.89 V
APPLE_TERM3	Termination 3 code: 2.89+ V

6.7.3.4 bc_charge_mode_t

```
enum bc_charge_mode_t
```

List of legacy battery charging schemes supported over a Type-C or Type-A port.

Enumerator

BC_CHARGE_NONE	No active battery charging modes.
BC_CHARGE_DCP	Dedicated Charging Port as defined by BC 1.2 spec.
BC_CHARGE_QC2	QC2.0 charger.
BC_CHARGE_QC3	QC3.0 charger.
BC_CHARGE_AFC	Adaptive Fast Charging mode.
BC_CHARGE_APPLE	Apple power brick.
BC_CHARGE_CDP	Charging Downstream Port as defined by BC 1.2 spec.

6.7.3.5 bc_d_status_t

```
enum bc_d_status_t
```

Enumeration of the various Dp/Dm states.

Enumerator

BC_D_GND	DP/DM voltage < 0.325
BC_D_0_6V	0.325 < DP/DPM voltage < 2V
BC_D_3_3V	DP/DM voltage > 2V
BC_D_ERR	Error state.

6.7.3.6 bc_port_role_t

```
enum bc_port_role_t
```

Battery charging port role.

Enumerator

BC_PORT_SINK	Power sink, device being charged.
BC_PORT_SOURCE	Power source.

6.7.3.7 bc_port_type_t

```
enum bc_port_type_t
```

Battery charging port types.

Enumerator

BC_PORT_TYPE_A	Type-A port. Only supported in power adapter and power bank designs.
BC_PORT_TYPE_C	Type-C port.

6.7.3.8 bc_qc_class_t

enum `bc_qc_class_t`

Qualcomm Quick Charge charger class.

Enumerator

BC_QC_CLASS↔ _A	Class A HVDCP: Supports 5V, 9V and 12V supplies.
BC_QC_CLASS↔ _B	Class B HVDCP: Supports 5V, 9B, 12V and 20V supplies.

6.7.3.9 bc_qc_ver_t

enum `bc_qc_ver_t`

Qualcomm charger version.

Enumerator

BC_QC_VER↔ _2	QC 2.0 charger.
BC_QC_VER↔ _3	QC 3.0 charger.

6.7.3.10 bc_sink_timer_t

enum `bc_sink_timer_t`

List of possible timers in sink mode.

Enumerator

BC_SINK_TIMER_NONE	No timers running.
BC_SINK_DCD_DEBOUNCE_TIMER	DCD Debounce timer.

6.7.3.11 bc_state_t

enum `bc_state_t`

List of states in the legacy battery charging state machine.

Enumerator

BC_FSM_OFF	BC state machine inactive.
BC_FSM_SRC_LOOK_FOR_CONNECT	Look for connection as a DCP.

Enumerator

BC_FSM_SRC_INITIAL_CONNECT	Initial BC1.2 sink connection detected.
BC_FSM_SRC_APPLE_CONNECTED	Connected to sink as an Apple power brick.
BC_FSM_SRC_CDP_CONNECTED	Port configured as a Charging Downstream Port (CDP).
BC_FSM_SRC_OTHERS_CONNECTED	DCP failed Apple sink detection.
BC_FSM_SRC_QC_OR_AFC	Port connected to a QC or AFC sink.
BC_FSM_SRC_QC_CONNECTED	Connected to sink as a QC HVDCP.
BC_FSM_SRC_AFC_CONNECTED	Connected to sink as an AFC charger.
BC_FSM_SINK_START	BC sink state machine start state.
BC_FSM_SINK_APPLE_CHARGER_DETECT	Sink looking for an Apple charger.
BC_FSM_SINK_APPLE_BRICK_ID_DETECT	Sink identified Apple charger, identifying brick ID.
BC_FSM_SINK_PRIMARY_CHARGER_DETECT	BC 1.2 primary charger detect state.
BC_FSM_SINK_TYPE_C_ONLY_SOURCE_CONNECTED	BC 1.2 src detection failed, connected as Type-C sink.
BC_FSM_SINK_SECONDARY_CHARGER_DETECT	BC 1.2 secondary charger detect state.
BC_FSM_SINK_DCP_CONNECTED	Sink connected to a BC 1.2 DCP.
BC_FSM_SINK_SDP_CONNECTED	Sink connected to a Standard Downstream Port (SDP).
BC_FSM_SINK_CDP_CONNECTED	Sink connected to a BC 1.2 CDP.
BC_FSM_MAX_STATES	Invalid state ID.

6.7.4 Function Documentation

6.7.4.1 bc_afc_form_vi()

```
void bc_afc_form_vi (
    uint8_t cport )
```

This function forms the new AFC source capabilities based on the power passed.

The number of source caps for AFC will be same as that configured in the table. This function does not modify the voltage value of the source cap in the config table. The current will be updated based on the power value and the AFC voltage.

Parameters

<i>cport</i>	Port index.
--------------	-------------

Returns

None

6.7.4.2 bc_clear_bc_evt()

```
void bc_clear_bc_evt (
```



```
uint8_t cport,
uint32_t evt_mask )
```

This function clears one or more BC events after the state machine has dealt with them.

Parameters

<i>cport</i>	Port index.
<i>evt_mask</i>	Event Mask to be cleared.

Returns

None

6.7.4.3 bc_fsm()

```
ccg_status_t bc_fsm (
uint8_t cport )
```

This function handles the Battery Charging block state machine.

Parameters

<i>cport</i>	Battery Charging port index.
--------------	------------------------------

Returns

ccg_status_t

6.7.4.4 bc_get_config()

```
const chg_cfg_params_t* bc_get_config (
uint8_t cport )
```

This function retrieves the current Battery Charging configuration. If the LEGACY_DYN_CFG_ENABLE macro is enabled, then this returns the latest configuration. If not, returns the configuration structure from the configuration table.

Parameters

<i>cport</i>	Battery Charging port index.
--------------	------------------------------

Returns

Pointer to current configuration data. This should not be modified by the caller.

6.7.4.5 bc_get_status()

```
const bc_status_t* bc_get_status (
```

```
uint8_t cport )
```

This function retrieves the current status of the BC state machine.

Parameters

<i>cport</i>	Battery Charging port index to be queried.
--------------	--

Returns

Pointer to BC status. The structure must not be modified by caller.

6.7.4.6 bc_init()

```
ccg_status_t bc_init (
    uint8_t cport )
```

This function initializes the Battery Charging block. This should be called one time only at system startup for each port on which the BC state machine needs to run.

Parameters

<i>cport</i>	Battery Charging port index.
--------------	------------------------------

Returns

`ccg_status_t`

6.7.4.7 bc_is_active()

```
bool bc_is_active (
    uint8_t cport )
```

This function returns whether the BC module is active or not.

Parameters

<i>cport</i>	Battery Charging port index.
--------------	------------------------------

Returns

true if the BC module is running, false otherwise.

6.7.4.8 bc_pd_event_handler()

```
void bc_pd_event_handler (
    uint8_t port,
    app_evt_t evt )
```

This function handles events from USB-PD Device Policy Manager. This event handler is used call the `bc_start` and `bc_stop` functions to enable/disable the BC state machine at appropriate times.

Parameters

<i>port</i>	PD port index.
<i>evt</i>	PD event.

Returns

None

6.7.4.9 `bc_port_is_cdp()`

```
bool bc_port_is_cdp (
    uint8_t cport )
```

Check whether the port is currently functioning as a CDP (Charging Downstream Port).

Parameters

<i>cport</i>	Battery Charging port index.
--------------	------------------------------

Returns

true if port is CDP, false otherwise.

6.7.4.10 `bc_set_bc_evt()`

```
void bc_set_bc_evt (
    uint8_t cport,
    uint32_t evt_mask )
```

This function sets an event status for the BC state machine to process.

Parameters

<i>cport</i>	Charging port index
<i>evt_mask</i>	Event Mask

Returns

None

6.7.4.11 `bc_sleep()`

```
bool bc_sleep (
    void )
```

This function puts the Battery Charging block to sleep.

Returns

Returns true if the sleep is successful, false otherwise.

6.7.4.12 bc_start()

```
ccg_status_t bc_start (
    uint8_t cport,
    bc_port_role_t port_role )
```

This function starts the Battery Charging block with desired configuration.

Parameters

<i>cport</i>	Battery Charging port index.
<i>port_role</i>	Battery Charging port power role.

Returns

ccg_status_t

6.7.4.13 bc_stop()

```
ccg_status_t bc_stop (
    uint8_t cport )
```

This function stops the Battery Charging block.

Parameters

<i>cport</i>	Battery Charging port index.
--------------	------------------------------

Returns

ccg_status_t

6.7.4.14 bc_wakeup()

```
bool bc_wakeup (
    void )
```

This function wakes up the Battery Charging block.

Returns

Returns true if wakeup successful, false otherwise

6.7.4.15 `ccg_get_system_max_pdp()`

```
uint8_t ccg_get_system_max_pdp (
    uint8_t cport )
```

This function returns the port PDP value.

If PD3.0 is supported, then the value returned will be the PDP value from the extended source cap. If PD3.0 is not supported, then the preconfigured PDP value will be returned.

Parameters

<code>cport</code>	Port index.
--------------------	-------------

Returns

System PDP

6.7.4.16 `qc_set_cf_limit()`

```
void qc_set_cf_limit (
    uint8_t cport )
```

This function enables Current Foldback on the port when in QC mode.

Parameters

<code>cport</code>	Port index.
--------------------	-------------

Returns

None

6.8 app/intel_tbt/bb_retimer.h File Reference

```
#include <pd.h>
```

Macros

- #define `BB_DEBUG_MODE_SIZE` (4u)
- #define `BB_DEBUGMODE_POLL_COUNT` (50u)
- #define `MAX_NO_OF_RETIMERS` (2u)
- #define `RETIMER_CFG_AVAILABLE` (0x80)
- #define `RETIMER_CFG_SLAVE_ADDR` (0x7F)
- #define `BB_CONN_STATE_REG` (0x04)
- #define `BB_STATUS_REG` (0x05)
- #define `BB_DEBUG_MODE_REG` (0x07)
- #define `BB_STATUS_MASK` (0xBEFFDFFFu)
- #define `BB_USB_3_SPEED_MASK` (0x00000040u)
- #define `BB_IRQ_HPD_MASK` (0x00004000u)
- #define `BB_STATUS_Sx_ACTIVE` (1 << 24)

- #define `BB_DEBUG_MODE_RX_LOCKED` (0x8000u)
- #define `BB_DEBUG_MODE_USB_COMPLIANCE` (0xD0000000u)
- #define `BB_DEBUG_MODE_CLEAR` (0u)
- #define `BB_DEBUG_MODE_DEFAULT` (0xFFFFFFFFu)
- #define `BB_DBR_DEBUG_POLL_DELAY` (100u)
- #define `BB_DBR_DEBUG_MODE_REG_WRITE_DELAY` (85u)
- #define `BB_DBR_WAKEUP_DELAY` (40u)
- #define `RETIMER_I2C_TIMEOUT` (0x64)

Enumerations

- enum `rt_evt_t` {
`RT_EVT_VSYS_REMOVED, RT_EVT_VSYS_ADDED, RT_EVT_RETRY_STATUS, RT_EVT_UPDATE_STATUS,`
`RT_EVT_FORCE_PWR_CHNG, RT_EVT_WRITE_DEBUG_MODE, RT_EVT_READ_DEBUG_MODE,`
`RT_MAX_EVTS` }

Functions

- void `retimer_init` (void)
- bool `retimer_is_present` (uint8_t port)
- void `retimer_enable` (uint8_t port, bool delay_enable)
- void `retimer_disable` (uint8_t port, bool delay_enable)
- void `set_retimer_status` (uint8_t port, uint8_t status)
- void `retimer_set_evt` (uint8_t port, rt_evt_t evt)
- void `retimer_clr_evt` (uint8_t port, rt_evt_t evt)
- void `retimer_start_debug_poll` (uint8_t port, uint32_t mode_data, uint8_t poll_count, uint8_t delay_enable)
- bool `retimer_status_update` (uint8_t port, uint32_t status, bool force_update)
- bool `retimer_update_is_pending` (uint8_t port)
- void `retimer_task` (uint8_t port)
- void `retimer_set_slave_address` (void)
- uint8_t `retimer_sleep_allowed` (void)
- void `retimer_set_compl_mode` (uint8_t compl_mode)
- void `retimer_force_enable` (uint8_t port)
- bool `retimer_reg_write` (uint8_t port, uint8_t reg, uint32_t value)
- bool `retimer_write_wrapper` (uint8_t slave_address, uint8_t offset, uint8_t *data, uint8_t num_bytes)
- bool `retimer_read_wrapper` (uint8_t slave_address, uint8_t offset, uint8_t *data_buffer, uint8_t num_bytes)

6.8.1 Detailed Description

Burnside/Delta bridge retimer interface definitions.

6.8.2 Macro Definition Documentation

6.8.2.1 BB_CONN_STATE_REG

```
#define BB_CONN_STATE_REG (0x04)
```

BB Register: Connection State.

6.8.2.2 BB_DBR_DEBUG_MODE_REG_WRITE_DELAY

```
#define BB_DBR_DEBUG_MODE_REG_WRITE_DELAY (85u)
```

Add a 85ms timer to delay the write to the debug mode register as part of the retimer enable sequence.

6.8.2.3 BB_DBR_DEBUG_POLL_DELAY

```
#define BB_DBR_DEBUG_POLL_DELAY (100u)
```

Add a 100ms timer to retry reading the DEBUG register from the retimer.

6.8.2.4 BB_DBR_WAKEUP_DELAY

```
#define BB_DBR_WAKEUP_DELAY (40u)
```

Retimer was in low power and NAKed the request. The initial I2C request will wake the retimer up after which it starts re-loading firmware. Give it ~40ms to do this and start accepting state change requests. (Intel retimers unload their firmware upon entering low power mode) Retry status update after 40ms. This is as per spec 614432-tgl-pd-ctrl-intfreq-burnsidebridgeretimer-prd-rev0p95 (sec 7.2)

6.8.2.5 BB_DEBUG_MODE_CLEAR

```
#define BB_DEBUG_MODE_CLEAR (0u)
```

BB Register : Clear.

6.8.2.6 BB_DEBUG_MODE_DEFAULT

```
#define BB_DEBUG_MODE_DEFAULT (0xFFFFFFFFu)
```

Default mask used to "AND" all debug mode data from retimers.

6.8.2.7 BB_DEBUG_MODE_REG

```
#define BB_DEBUG_MODE_REG (0x07)
```

BB Register: Debug Mode.

6.8.2.8 BB_DEBUG_MODE_RX_LOCKED

```
#define BB_DEBUG_MODE_RX_LOCKED (0x8000u)
```

'Rx Locked' bit in the status register.

6.8.2.9 BB_DEBUG_MODE_SIZE

```
#define BB_DEBUG_MODE_SIZE (4u)
```

Size of the BB Debug Mode register.

6.8.2.10 BB_DEBUG_MODE_USB_COMPLIANCE

```
#define BB_DEBUG_MODE_USB_COMPLIANCE (0xD0000000u)
```

Bits to be set in DEBUG_MODE for USB Compliance mode entry Bit 31 (Global Enable) = 1 Bit 30 (Compliance Enabled) = 1 Bit 29 (USB Compliance Mode) = 1 Bit 28 (HTI Mode) = 0

6.8.2.11 BB_DEBUGMODE_POLL_COUNT

```
#define BB_DEBUGMODE_POLL_COUNT (50u)
```

Number of times to poll the retimer before giving up.

6.8.2.12 BB_IRQ_HPD_MASK

```
#define BB_IRQ_HPD_MASK (0x00004000u)
```

Bit 14 IRQ HPD Mask.

6.8.2.13 BB_STATUS_MASK

```
#define BB_STATUS_MASK (0xBEFFDFFFu)
```

TBT Status bits that are used in BB.

6.8.2.14 BB_STATUS_REG

```
#define BB_STATUS_REG (0x05)
```

BB Register: TBT Status.

6.8.2.15 BB_STATUS_Sx_ACTIVE

```
#define BB_STATUS_Sx_ACTIVE (1 << 24)
```

Bit 30 (previously reserved) indicates S0=0 vs Sx=1.

6.8.2.16 BB_USB_3_SPEED_MASK

```
#define BB_USB_3_SPEED_MASK (0x00000040u)
```

Bit 6 USB3 Speed Mask.

6.8.2.17 MAX_NO_OF_RETIMERS

```
#define MAX_NO_OF_RETIMERS (2u)
```

Max number of retimers available on any platform.

6.8.2.18 RETIMER_CFG_AVAILABLE

```
#define RETIMER_CFG_AVAILABLE (0x80)
```

Retimer Configuration bit: is retimer available?.

6.8.2.19 RETIMER_CFG_SLAVE_ADDR

```
#define RETIMER_CFG_SLAVE_ADDR (0x7F)
```

Retimer Configuration bit: 7-bit slave address.

6.8.2.20 RETIMER_I2C_TIMEOUT

```
#define RETIMER_I2C_TIMEOUT (0x64)
```

Timeout for the function completion is 100 ms.

6.8.3 Enumeration Type Documentation

6.8.3.1 rt_evt_t

```
enum rt_evt_t
```

List of retimer related events to be handled by the CCGx firmware.

Enumerator

RT_EVT_VSYS_REMOVED	VSYS supply to CCGx device removed. Need to disable retimer.
RT_EVT_VSYS_ADDED	VSYS supply to CCGx device restored.
RT_EVT_RETRY_STATUS	Retimer status update has to be retried.
RT_EVT_UPDATE_STATUS	Retimer status has to be changed.
RT_EVT_FORCE_PWR_CHNG	Force retimer power ON requested.
RT_EVT_WRITE_DEBUG_MODE	Write to retimer debug register requested.
RT_EVT_READ_DEBUG_MODE	Read from retimer debug register requested.
RT_MAX_EVTS	Invalid event.

6.8.4 Function Documentation

6.8.4.1 retimer_clr_evt()

```
void retimer_clr_evt (
    uint8_t port,
    rt_evt_t evt )
```

Clear an event on the retimer module.

Parameters

<i>port</i>	Port index the function is performed for.
<i>evt</i>	Event to clear on the module

Returns

None

6.8.4.2 retimer_disable()

```
void retimer_disable (
```

```
uint8_t port,
bool delay_enable )
```

Disable retimer on the specific port. This holds its reset and turns the power gate off.

Parameters

<i>port</i>	Port index the function is performed for.
<i>delay_enable</i>	Flag that indicates that disable sequence should be delayed.

Returns

None

6.8.4.3 retimer_enable()

```
void retimer_enable (
    uint8_t port,
    bool delay_enable )
```

Enable retimer on the specific port. This releases its reset and turns the power gate on.

Parameters

<i>port</i>	Port index the function is performed for.
<i>delay_enable</i>	Flag that indicates that enable sequence should be delayed.

Returns

None

6.8.4.4 retimer_force_enable()

```
void retimer_force_enable (
    uint8_t port )
```

Enable retimer on the specific port.

Parameters

<i>port</i>	Port index the function is performed for.
-------------	---

Returns

None

6.8.4.5 retimer_init()

```
void retimer_init (
```

```
void )
```

This initializes the retimer I2C and GPIO interfaces.

Returns

None

6.8.4.6 retimer_is_present()

```
bool retimer_is_present (
    uint8_t port )
```

Function check whether retimers are present on the corresponding Type-C port.

Parameters

<i>port</i>	Type-C port index.
-------------	--------------------

Returns

True if there is at least one retimer associated with the port.

6.8.4.7 retimer_read_wrapper()

```
bool retimer_read_wrapper (
    uint8_t slave_address,
    uint8_t offset,
    uint8_t * data_buffer,
    uint8_t num_bytes )
```

Wrapper for performing reads using the RTM PSoC Creator Component.

Parameters

<i>slave_address</i>	Address of the slave
<i>offset</i>	Offset in the slave to read from
<i>data_buffer</i>	Pointer to the buffer into which data is read
<i>num_bytes</i>	Number of bytes to be read

Returns

True if read was successful otherwise False

6.8.4.8 retimer_reg_write()

```
bool retimer_reg_write (
    uint8_t port,
    uint8_t reg,
    uint32_t value )
```

Write to the retimer register.

Parameters

<i>port</i>	Port index the function is performed for.
<i>reg</i>	Register the function is performed for.
<i>value</i>	Register value for update.

Returns

None

6.8.4.9 retimer_set_compl_mode()

```
void retimer_set_compl_mode (
    uint8_t compl_mode )
```

Indicator of whether host is in compliance mode.

Parameters

<i>compl_mode</i>	Mode status
-------------------	-------------

Returns

None

6.8.4.10 retimer_set_evt()

```
void retimer_set_evt (
    uint8_t port,
    rt_evt_t evt )
```

Set an event on the retimer module.

Parameters

<i>port</i>	Port index the function is performed for.
<i>evt</i>	Event to signal on the module

Returns

None

6.8.4.11 retimer_set_slave_address()

```
void retimer_set_slave_address (
    void )
```

Set the retimer slave address.

Returns

None

6.8.4.12 retimer_sleep_allowed()

```
uint8_t retimer_sleep_allowed (
    void )
```

Check if retimer interface can go to sleep.

Returns

bool Can enter sleep or not

6.8.4.13 retimer_start_debug_poll()

```
void retimer_start_debug_poll (
    uint8_t port,
    uint32_t mode_data,
    uint8_t poll_count,
    uint8_t delay_enable )
```

Start polling the debug mode register on the retimer.

Parameters

<i>port</i>	Port index the function is performed for.
<i>mode_data</i>	Debug mode data
<i>poll_count</i>	Number of polling attempts
<i>delay_enable</i>	Delay the debug mode register write

Returns

None

6.8.4.14 retimer_status_update()

```
bool retimer_status_update (
    uint8_t port,
    uint32_t status,
    bool force_update )
```

Update the retimer status.

Parameters

<i>port</i>	Port index the function is performed for.
<i>status</i>	Retimer status

Parameters

<i>force_update</i>	Force update of retimer status
---------------------	--------------------------------

Returns

bool Result of the update

6.8.4.15 retimer_task()

```
void retimer_task (
    uint8_t port )
```

Run the retimer task for that port.

Parameters

<i>port</i>	Port index the retimer functions are handled for
-------------	--

Returns

None

6.8.4.16 retimer_update_is_pending()

```
bool retimer_update_is_pending (
    uint8_t port )
```

Check if retimer update is pending.

Parameters

<i>port</i>	Port index the function is performed for.
-------------	---

Returns

bool Update pending status

6.8.4.17 retimer_write_wrapper()

```
bool retimer_write_wrapper (
    uint8_t slave_address,
    uint8_t offset,
    uint8_t * data,
    uint8_t num_bytes )
```

Wrapper for performing writes using the RTM PSoC Creator Component.

Parameters

<i>slave_address</i>	Address of the slave
<i>offset</i>	Offset in the slave to write to
<i>data</i>	Pointer to the buffer containing data to be written to the slave
<i>num_bytes</i>	Number of bytes to write

Returns

True if write was successful otherwise False

6.8.4.18 set_retimer_status()

```
void set_retimer_status (
    uint8_t port,
    uint8_t status )
```

Set the status of retimer.

Parameters

<i>port</i>	PD port associated with the retimer to be updated.
<i>status</i>	Enable or disable flag for retimer.

Returns

None

6.9 app/intel_tbt/icl.h File Reference

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

Macros

- #define ICL_CTRL_CMD_FORCE_TBT_MODE (0x01u)
- #define ICL_CTRL_CMD_SOC_TO_EVT_DISABLE (0x80u)
- #define ICL_STS_REG_FORCE_TBT_MODE (1 << 0)
- #define ICL_RFU_EXIT_DURATION (12u)
- #define ICL_ADP_DETACH_DEBOUNCE_TIME (100)
- #define ICL_ADP_ATTACH_DEBOUNCE_TIME (5)
- #define ADP_STATE_INIT (2)
- #define ADP_NEGEDGE (0)
- #define ADP_POSEDGE (1)
- #define ADP_EDGE_NONE (2)

Enumerations

- enum `icl_reg_t` {
`ICL_CTRL_REG = 0x0040, ICL_STS_REG = 0x0042, ICL_DBG_REG = 0x0044, ICL_BB_RETIMER_FORCE_PWR = 0x0045,`
`ICL_BB_RETIMER_CMD_REG = 0x0046, ICL_BB_RETIMER_DAT_REG = 0x0048 }`
- enum `icl_evt_t` { `ICL_EVT_DEQUEUE_HPD, ICL_EVT_CHANGE_MUX_STATE, ICL_MAX_EVTS` }

Functions

- void `icl_init` (void)
- void `icl_task` (uint8_t port)
- void `icl_set_evt` (uint8_t port, `icl_evt_t` evt)
- bool `icl_vsys_is_present` (void)
- bool `icl_sleep_allowed` (void)

6.9.1 Detailed Description

Ice Lake specific logic.

6.9.2 Macro Definition Documentation

6.9.2.1 ADP_EDGE_NONE

```
#define ADP_EDGE_NONE (2)
```

No transition seen on DC barrel presence detect pin.

6.9.2.2 ADP_NEGEDGE

```
#define ADP_NEGEDGE (0)
```

Negative edge seen on DC barrel presence detect pin.

6.9.2.3 ADP_POSEDGE

```
#define ADP_POSEDGE (1)
```

Positive edge seen on DC barrel presence detect pin.

6.9.2.4 ADP_STATE_INIT

```
#define ADP_STATE_INIT (2)
```

Initial state of the DC barrel adapter detect input. Setting to a value other than 0 and 1.

6.9.2.5 ICL_ADP_ATTACH_DEBOUNCE_TIME

```
#define ICL_ADP_ATTACH_DEBOUNCE_TIME (5)
```

Debounce time in ms used to confirm that DC barrel adapter has been connected.

6.9.2.6 ICL_ADP_DETACH_DEBOUNCE_TIME

```
#define ICL_ADP_DETACH_DEBOUNCE_TIME (100)
```

Debounce time in ms used to confirm that DC barrel adapter has been removed.

6.9.2.7 ICL_CTRL_CMD_FORCE_TBT_MODE

```
#define ICL_CTRL_CMD_FORCE_TBT_MODE (0x01u)
```

HPI command code to trigger retimer firmware update mode.

6.9.2.8 ICL_CTRL_CMD_SOC_TO_EVT_DISABLE

```
#define ICL_CTRL_CMD_SOC_TO_EVT_DISABLE (0x80u)
```

HPI register bit to disable SoC Ack Timeout event.

6.9.2.9 ICL_RFU_EXIT_DURATION

```
#define ICL_RFU_EXIT_DURATION (12u)
```

Delay in ms before Retimer is brought out of firmware update mode.

6.9.2.10 ICL_STS_REG_FORCE_TBT_MODE

```
#define ICL_STS_REG_FORCE_TBT_MODE (1 << 0)
```

HPI status register bit that indicates retimer has been forced into USB4/TBT mode.

6.9.3 Enumeration Type Documentation

6.9.3.1 icl_evt_t

```
enum icl_evt_t
```

List of IceLake platform specific events to be handled by the CCG firmware.

Enumerator

ICL_EVT_DEQUEUE_HPD	0x00 : ICL Dequeue HPD event
ICL_EVT_CHANGE_MUX_STATE	0x01 : ICL MUX change state event
ICL_MAX_EVTS	0x02 : ICL Maximum events

6.9.3.2 icl_reg_t

```
enum icl_reg_t
```

List of HPI control registers supported for IceLake platform management.

Enumerator

ICL_CTRL_REG	ICL Control Register
ICL_STS_REG	ICL Status Register
ICL_DBG_REG	ICL Debug Register
ICL_BB_RETIMER_FORCE_PWR	Register to force power to retimer.
ICL_BB_RETIMER_CMD_REG	ICL HPI Retimer Command register 45:47h
ICL_BB_RETIMER_DAT_REG	ICL HPI Retimer Data Register 48:4Bh

6.9.4 Function Documentation

6.9.4.1 icl_init()

```
void icl_init (
    void )
```

Initialize logic needed for Ice Lake designs.

Returns

None

6.9.4.2 icl_set_evt()

```
void icl_set_evt (
    uint8_t port,
    icl_evt_t evt )
```

Set an event on the ICL module.

Parameters

<i>port</i>	Port index
<i>evt</i>	ICL event

Returns

None

6.9.4.3 icl_sleep_allowed()

```
bool icl_sleep_allowed (
    void )
```

Does ICL module allow CCG to enter deep-sleep.

Returns

bool

6.9.4.4 icl_task()

```
void icl_task (
    uint8_t port )
```

Run ICL RVP-specific tasks.

Parameters

<i>port</i>	Port index
-------------	------------

Returns

None

6.9.4.5 icl_vsys_is_present()

```
bool icl_vsys_is_present (
    void )
```

Helper function to get VSYS state.

Returns

bool True if VSYS is available else False

6.10 app/intel_tbt/intel_ridge.h File Reference

```
#include <alt_mode_hw.h>
#include <status.h>
#include <hpd.h>
```

Functions

- bool [ridge_set_mux](#) (uint8_t port, [mux_select_t](#) mux_cfg, uint8_t polarity, uint32_t cfg)
- [ccg_status_t](#) [tr_hpd_init](#) (uint8_t port, [hpd_event_cbk_t](#) cbk)
- void [tr_hpd_deinit](#) (uint8_t port)
- [ccg_status_t](#) [tr_hpd_sendevt](#) (uint8_t port, [hpd_event_type_t](#) evtype)
- void [ridge_eval_cmd](#) (uint8_t port, uint32_t stat, uint32_t stat_mask)
- bool [tr_is_hpd_change](#) (uint8_t port)
- void [ridge_set_vpro](#) (uint8_t port, bool en_status)
- void [ridge_update_dr](#) (uint8_t port)
- void [ridge_force_status_update](#) (uint8_t port, uint8_t force_update)

6.10.1 Detailed Description

Intel Alpine/Titan Ridge control interface header file.

6.10.2 Function Documentation

6.10.2.1 `ridge_eval_cmd()`

```
void ridge_eval_cmd (
    uint8_t port,
    uint32_t stat,
    uint32_t stat_mask )
```

Analyses received ridge command register content.

Parameters

<i>port</i>	USB-PD port index corresponding to the status update.
<i>stat</i>	Command register value.
<i>stat_mask</i>	Bit mask of TR command register which were changed from the previous time

Returns

true if the interface is idle, false otherwise.

6.10.2.2 `ridge_force_status_update()`

```
void ridge_force_status_update (
    uint8_t port,
    uint8_t force_update )
```

Force TBT status update.

Parameters

<i>port</i>	PD port index.
<i>force_update</i>	Force next update on this port

Returns

None

6.10.2.3 `ridge_set_mux()`

```
bool ridge_set_mux (
    uint8_t port,
    mux_select_t mux_cfg,
```

```
uint8_t polarity,
uint32_t cfg )
```

This function set AR/TR registers in accordance to input parameters.

Parameters

<i>port</i>	Port index the AR/TR settings are performed for.
<i>mux_cfg</i>	MUX configuration.
<i>polarity</i>	Attached target Type-C Polarity.
<i>cfg</i>	Contains AR/TR register settings in case of TBT alt mode is active.

Returns

true if AR/TR was set successful, in the other case - false

6.10.2.4 ridge_set_vpro()

```
void ridge_set_vpro (
    uint8_t port,
    bool en_status )
```

Enable/Disable vPro register in ridge IC.

Parameters

<i>port</i>	USB-PD port index corresponding to the vPro mode.
<i>en_status</i>	Enable or disable flag for vPro mode.

Returns

None.

6.10.2.5 ridge_update_dr()

```
void ridge_update_dr (
    uint8_t port )
```

Updates data role bit in ridge IC.

Parameters

<i>port</i>	PD port index.
-------------	----------------

Returns

None.

6.10.2.6 tr_hpd_deinit()

```
void tr_hpd_deinit (
    uint8_t port )
```

Disables Titan Ridge the HPD functionality for the specified PD port.

Parameters

<i>port</i>	PD port index. Caller should ensure to provide only valid values.
-------------	---

Returns

None.

6.10.2.7 tr_hpd_init()

```
ccg_status_t tr_hpd_init (
    uint8_t port,
    hpd_event_cbk_t cbk )
```

Enables Titan Ridge the HPD functionality for the specified PD port.

Parameters

<i>port</i>	PD port index. Caller should ensure to provide only valid values.
<i>cbk</i>	callback to be used for command completion event.

Returns

Returns CCG_STAT_SUCCESS in case of success, error code otherwise.

6.10.2.8 tr_hpd_sendevt()

```
ccg_status_t tr_hpd_sendevt (
    uint8_t port,
    hpd_event_type_t evtype )
```

Send the desired HPD event out through the Titan Ridge HPD GPIO. Only the HPD_EVENT_UNPLUG, HPD_EVENT_UNPLUG and HPD_EVENT_IRQ events should be requested.

Parameters

<i>port</i>	Port on which HPD event is to be sent.
<i>evtype</i>	Type of HPD event to be sent.

Returns

Returns CCG_STAT_SUCCESS in case of success, error code otherwise.

6.10.2.9 tr_is_hpd_change()

```
bool tr_is_hpd_change (
    uint8_t port )
```

Indicates is HPD level changed from the previous state.

Parameters

<i>port</i>	USB-PD port index corresponding to the status update.
-------------	---

Returns

None.

6.11 app/intel_tbt/intel_vid.h File Reference

```
#include <pd.h>
#include <alt_modes_mngr.h>
```

Macros

- #define INTEL_VID (0x8087u)
- #define TBT_ALT_MODE_ID (1u)
- #define VPRO_ALT_MODE_ID (2u)
- #define MAX_TBT_VDO_NUMB (1u)
- #define TBT_VDO_IDX (0u)
- #define GET_LEGACY_TBT_ADAPTER(status) ((status >> 16) & 0x1)
- #define TBT_EXIT(status) ((status >> 4) & 0x1)
- #define BB_STATUS(status) ((status >> 3) & 0x1)
- #define USB2_ENABLE(status) ((status >> 2) & 0x1)
- #define BB_STS_USB2_ENABLE(status) ((status >> 2) & 0x3)
- #define BB_PRESENT_EXIT_MODE (0x03u)

Enumerations

- enum tbt_state_t { TBT_STATE_IDLE = 0, TBT_STATE_ENTER = 4, TBT_STATE_EXIT = 5, TBT_STATE_ATT = 6 }
- enum tbt_cbl_speed_t { TBT_CBL_SPEED_GEN_1 = 1, TBT_CBL_SPEED_10_GB = 2, TBT_CBL_SPEED_10_20_GB = 3 }
- enum tbt_cbl_gen_t { TBT_CBL_GEN_3 = 0, TBT_CBL_GEN_4 = 1 }

Functions

- alt_mode_info_t * reg_intel_modes (uint8_t port, alt_mode_reg_info_t *reg_info)

6.11.1 Detailed Description

Thunderbolt (Intel VID) alternate mode handler header file.

6.11.2 Macro Definition Documentation

6.11.2.1 BB_STATUS

```
#define BB_STATUS(  
    status ) ((status >> 3) & 0x1)
```

Macro to get BB status from Attention VDO.

6.11.2.2 GET_LEGACY_TBT_ADAPTER

```
#define GET_LEGACY_TBT_ADAPTER(  
    status ) ((status >> 16) & 0x1)
```

Macro to get legacy adapter status from Discovery Mode response VDO.

6.11.2.3 INTEL_VID

```
#define INTEL_VID (0x8087u)
```

Intel (Thunderbolt 3) SVID.

6.11.2.4 MAX_TBT_VDO_NUMB

```
#define MAX_TBT_VDO_NUMB (1u)
```

Maximum number of VDOs Thunderbolt-3 uses for the alt mode flow.

6.11.2.5 TBT_ALT_MODE_ID

```
#define TBT_ALT_MODE_ID (1u)
```

Unique ID assigned to Thunderbolt mode in the CCGx SDK.

6.11.2.6 TBT_EXIT

```
#define TBT_EXIT(  
    status ) ((status >> 4) & 0x1)
```

Macro to get Exit status from Attention VDO.

6.11.2.7 TBT_VDO_IDX

```
#define TBT_VDO_IDX (0u)
```

Index of VDO used for Thunderbolt.

6.11.2.8 USB2_ENABLE

```
#define USB2_ENABLE(  
    status ) ((status >> 2) & 0x1)
```

Macro to get USB enable status from Attention VDO.

6.11.2.9 VPRO_ALT_MODE_ID

```
#define VPRO_ALT_MODE_ID (2u)
```

Unique ID assigned to Vpro mode in the CCGx SDK.

6.11.3 Enumeration Type Documentation

6.11.3.1 tbt_cbl_gen_t

```
enum tbt_cbl_gen_t
```

This enumeration holds all possible TBT cable generations.

Enumerator

TBT_CBL_GEN↔ _3	3rd generation TBT (10.3125 and 20.625 Gb/s).
TBT_CBL_GEN↔ _4	4th generation TBT (10.0, 10.3125, 20.0 and 20.625 Gb/s).

6.11.3.2 tbt_cbl_speed_t

```
enum tbt_cbl_speed_t
```

This enumeration holds all possible TBT cable speeds.

Enumerator

TBT_CBL_SPEED_GEN_1	USB3.1 gen1 cable (10Gb/s TBT support).
TBT_CBL_SPEED_10_GB	10Gb/s.
TBT_CBL_SPEED_10_20_GB	10Gb/s and 20Gb/s.

6.11.3.3 tbt_state_t

```
enum tbt_state_t
```

This enumeration holds all possible TBT states.

Enumerator

TBT_STATE_IDLE	Idle state.
TBT_STATE_ENTER	Enter mode state.
TBT_STATE_EXIT	Exit mode state.
TBT_STATE_ATT	Attention state.

6.11.4 Function Documentation

6.11.4.1 `reg_intel_modes()`

```
alt_mode_info_t* reg_intel_modes (
    uint8_t port,
    alt_mode_reg_info_t * reg_info )
```

This function analyses Discovery information to find out if further TBT alternative mode processing is allowed.

Parameters

<i>port</i>	Port index the function is performed for.
<i>reg_info</i>	Pointer to structure which holds alt mode register info.

Returns

Pointer to TBT alternative mode command structure if analysis passed successful. In case of failure, function returns NULL pointer.

6.12 `app/intel_tbt/ridge_slave.h` File Reference

```
#include <stdbool.h>
#include <stdint.h>
#include "config.h"
#include "status.h"
#include "i2c.h"
```

Macros

- `#define RIDGE_SLAVE_SCB_INDEX` (0x1)
- `#define RIDGE_SLAVE_SCB_CLOCK_FREQ` (I2C_SCB_CLOCK_FREQ_1_MHZ)
- `#define RIDGE_SLAVE_MIN_WRITE_SIZE` (2)
- `#define RIDGE_SLAVE_MAX_WRITE_SIZE` (16)
- `#define RIDGE_SLAVE_MAX_READ_SIZE` (16)
- `#define PMC_SLAVE_ADDR_PORT0` (0x50)
- `#define PMC_SLAVE_ADDR_PORT1` (0x51)
- `#define ICL_SOC_ACK_TIMEOUT_PERIOD` (500u)
- `#define RIDGE_SLAVE_ADDR_MASK_GENERAL` (0x80)
- `#define RIDGE_CMD_CCG_RESET` (0x02)
- `#define RIDGE_CMD_INT_CLEAR` (0x04)
- `#define RIDGE_IRQ_ACK` (0x2000)

Enumerations

- enum `ridge_slave_reg_addr_t` { `RIDGE_REG_CONTROLLER_STATE` = 0x03, `RIDGE_REG_CCG_COMMAND` = 0x50, `RIDGE_REG_CCG_STATUS` = 0x5F, `RIDGE_REG_RETIMER_DEBUG` = 0x5D }
- enum `irq_state_t` { `DFP_IRQ_DEFAULT_STATE`, `DFP_IRQ_SENT_TO_TR`, `DFP_IRQ_CLR_BY_CCG`, `DFP_IRQ_CLR_ACK_BY_TR` }

Functions

- void [ridge_slave_init](#) (uint8_t scbnum, uint8_t portsel)
- void [ridge_slave_deinit](#) (void)
- void [ridge_slave_status_update](#) (uint8_t port, uint32_t status, bool rewrite)
- void [ridge_slave_task](#) (void)
- bool [ridge_slave_sleep](#) (void)
- void [ridge_reg_reset](#) (uint8_t port)
- uint8_t [ridge_update_is_pending](#) (void)
- bool [ridge_slave_is_host_connected](#) (uint8_t port)
- void [ridge_slave_set_ocp_status](#) (uint8_t port)
- void [ridge_slave_soc_timeout_event_control](#) (bool disable)

6.12.1 Detailed Description

Alpine/Titan Ridge I2C slave interface header file.

6.12.2 Macro Definition Documentation

6.12.2.1 ICL_SOC_ACK_TIMEOUT_PERIOD

```
#define ICL_SOC_ACK_TIMEOUT_PERIOD (500u)
```

t_SOCAckTimeout period defined in PD controller specs from Intel.

6.12.2.2 PMC_SLAVE_ADDR_PORT0

```
#define PMC_SLAVE_ADDR_PORT0 (0x50)
```

Port 0 slave default slave address.

6.12.2.3 PMC_SLAVE_ADDR_PORT1

```
#define PMC_SLAVE_ADDR_PORT1 (0x51)
```

Port 1 slave default slave address.

6.12.2.4 RIDGE_CMD_CCG_RESET

```
#define RIDGE_CMD_CCG_RESET (0x02)
```

Alpine/Titan Ridge command value that requests a CCG device reset.

6.12.2.5 RIDGE_CMD_INT_CLEAR

```
#define RIDGE_CMD_INT_CLEAR (0x04)
```

Alpine/Titan Ridge command value to clear the interrupt from CCG.

6.12.2.6 RIDGE_IRQ_ACK

```
#define RIDGE_IRQ_ACK (0x2000)
```

Alpine/Titan Ridge command IRQ ACK PD Controller to Titan Ridge.

6.12.2.7 RIDGE_SLAVE_MAX_READ_SIZE

```
#define RIDGE_SLAVE_MAX_READ_SIZE (16)
```

Maximum slave read size.

6.12.2.8 RIDGE_SLAVE_MAX_WRITE_SIZE

```
#define RIDGE_SLAVE_MAX_WRITE_SIZE (16)
```

Maximum slave write size: We should never get writes longer than 16 bytes.

6.12.2.9 RIDGE_SLAVE_MIN_WRITE_SIZE

```
#define RIDGE_SLAVE_MIN_WRITE_SIZE (2)
```

Minimum slave write size: Corresponds to slave address + register address.

6.12.2.10 RIDGE_SLAVE_SCB_CLOCK_FREQ

```
#define RIDGE_SLAVE_SCB_CLOCK_FREQ (I2C_SCB_CLOCK_FREQ_1_MHZ)
```

Maximum I2C clock frequency used on the Ridge/SoC slave interface.

6.12.2.11 RIDGE_SLAVE_SCB_INDEX

```
#define RIDGE_SLAVE_SCB_INDEX (0x1)
```

Default SCB index used for the Ridge/SoC Slave interface.

6.12.3 Enumeration Type Documentation

6.12.3.1 `ridge_slave_reg_addr_t`

```
enum ridge_slave_reg_addr_t
```

List of Alpine/Titan Ridge slave interface registers.

The Thunderbolt Alternate Mode specification defines the following set of registers that should be implemented by a USB-PD port controller in Thunderbolt enabled systems.

Enumerator

<code>RIDGE_REG_CONTROLLER_STATE</code>	PD controller state register.
<code>RIDGE_REG_CCG_COMMAND</code>	Data Control register.
<code>RIDGE_REG_CCG_STATUS</code>	Data Status register.
<code>RIDGE_REG_RETIMER_DEBUG</code>	Retimer debug register

6.12.4 Function Documentation

6.12.4.1 `ridge_reg_reset()`

```
void ridge_reg_reset (
    uint8_t port )
```

Resets Ridge related registers.

Parameters

<i>port</i>	USB-PD port index corresponding to the register reset.
-------------	--

Returns

None

6.12.4.2 `ridge_slave_deinit()`

```
void ridge_slave_deinit (
    void )
```

De-initialize the Alpine/Titan Ridge slave interface module.

This function de-initializes the SCB block used to implement the I2C slave interface between CCGx and Alpine/Titan Ridge.

6.12.4.3 `ridge_slave_init()`

```
void ridge_slave_init (
    uint8_t scbnum,
    uint8_t portsel )
```

Initialize the Alpine/Titan Ridge slave interface module.

This function initializes the Alpine/Titan Ridge slave interface module and configures it to use the specified SCB block. The SCB will be configured as an I2C slave block, and the interrupt output will also be initialized to a de-asserted state.

Since only two registers are to be implemented, and the commands to be implemented are simple, the complete module is implemented using the I2C command callbacks.

Parameters

<i>scbnum</i>	SCB index to be used for the Alpine/Titan Ridge slave interface.
<i>portsel</i>	Alpine/Titan Ridge slave port selection. Only applicable for single port designs.

Returns

None

6.12.4.4 `ridge_slave_is_host_connected()`

```
bool ridge_slave_is_host_connected (
    uint8_t port )
```

Check whether a host is connected to the specified PD port.

Parameters

<i>port</i>	Index of the PD port.
-------------	-----------------------

Returns

true if a host is connected, false otherwise.

6.12.4.5 `ridge_slave_set_ocp_status()`

```
void ridge_slave_set_ocp_status (
    uint8_t port )
```

Update the data status register to indicate OverCurrent fault condition.

Parameters

<i>port</i>	PD port index.
-------------	----------------

Returns

None

6.12.4.6 `ridge_slave_sleep()`

```
bool ridge_slave_sleep (
    void )
```

Check whether the AR/TR slave interface is idle so that device can be placed into sleep.

This function should be called prior to placing the CCG device in deep sleep. Deep sleep entry is only allowed if this function returns true.

Returns

true if the interface is idle, false otherwise.

6.12.4.7 `ridge_slave_soc_timeout_event_control()`

```
void ridge_slave_soc_timeout_event_control (
    bool disable )
```

Enable or disable HPI notifications about SOC Ack timeout condition. This API is only valid on Ice Lake / Tiger Lake platforms and will be a NOP otherwise.

Parameters

<i>disable</i>	Whether HPI notification on SOC Ack timeout should be disabled.
----------------	---

Returns

None

6.12.4.8 ridge_slave_status_update()

```
void ridge_slave_status_update (
    uint8_t port,
    uint32_t status,
    bool rewrite )
```

Update the AR/TR status register and send an event to the Alpine/Titan Ridge.

This function is used by the application layer to update the content of the Alpine/Titan Ridge status register. If the content of the register is changing, CCG asserts the corresponding interrupt to notify Alpine/Titan Ridge about the status change.

Parameters

<i>port</i>	USB-PD port index corresponding to the status update.
<i>status</i>	Value to be written into the status register.
<i>rewrite</i>	Flag to enable updating of the status register even it remains the same.

Returns

None

6.12.4.9 ridge_slave_task()

```
void ridge_slave_task (
    void )
```

Handler for pending Ridge Slave interface tasks.

Returns

None

6.12.4.10 ridge_update_is_pending()

```
uint8_t ridge_update_is_pending (
    void )
```

Checks if any PMC/Ridge update is pending.

Returns

uint8_t Bitmask of updates pending

6.13 app/pdo.h File Reference

```
#include <pd.h>
```

Macros

- #define `APP_PPS_SNK_CONTRACT_PERIOD` (9000u)
- #define `APP_PPS_SNK_CONTRACT_RETRY_PERIOD` (5u)

Functions

- void `eval_src_cap` (uint8_t port, const `pd_packet_t` *src_cap, `app_resp_cbk_t` app_resp_handler)
- void `eval_rdo` (uint8_t port, `pd_do_t` rdo, `app_resp_cbk_t` app_resp_handler)
- void `app_update_rdo` (uint8_t port, const `pd_packet_t` *src_cap, `app_resp_t` *app_resp)

6.13.1 Detailed Description

PDO evaluation and handler definitions.

6.13.2 Macro Definition Documentation

6.13.2.1 APP_PPS_SNK_CONTRACT_PERIOD

```
#define APP_PPS_SNK_CONTRACT_PERIOD (9000u)
```

Period after which a PPS Sink repeats PD contract attempts. This should be faster than once in 10 seconds.

6.13.2.2 APP_PPS_SNK_CONTRACT_RETRY_PERIOD

```
#define APP_PPS_SNK_CONTRACT_RETRY_PERIOD (5u)
```

Period after which a failed PPS sink re-contract attempt will be retried.

6.13.3 Function Documentation

6.13.3.1 app_update_rdo()

```
void app_update_rdo (  
    uint8_t port,  
    const pd_packet_t * src_cap,  
    app_resp_t * app_resp )
```


This function is only applicable in the case of the CCG6 device, and can be used by the application logic to modify the RDO generated by the ROM-ed version of the eval_src_cap function.

Parameters

<i>port</i>	Port index the function is performed for.
<i>src_cap</i>	Pointer to PD packet which contains source capabilities.
<i>app_resp</i>	Response containing the updated RDO.

Returns

None

6.13.3.2 eval_rdo()

```
void eval_rdo (
    uint8_t port,
    pd_do_t rdo,
    app_resp_cbk_t app_resp_handler )
```

This function is called by the PD stack to allow the application to evaluate a power request data object received from the port partner and decide whether it should be satisfied. The response to the request should be passed back to the stack through the `app_resp_handler()` callback.

Warning

On the CCG6 device, the `eval_rdo` function is executed from ROM and hence changes in this function will not affect the policy engine behavior. The application is expected to update the source capabilities registered with the PD stack based on the current system status.

Parameters

<i>port</i>	Port index the function is performed for.
<i>rdo</i>	The request data object received.
<i>app_resp_handler</i>	Application handler callback function.

Returns

None

6.13.3.3 eval_src_cap()

```
void eval_src_cap (
    uint8_t port,
    const pd_packet_t * src_cap,
    app_resp_cbk_t app_resp_handler )
```

This function is called by the PD stack to allow the application logic to evaluate the Source Capabilities received from the port partner and generate the desired request. The request object is expected to be passed back to the stack through the `app_resp_handler()` callback.

The default implementation of this function matches each of the received source PDOs against the active sink capabilities; and then selects the source PDO that can deliver the maximum power to the system as a sink.

Warning

On the CCG6 device, the `eval_src_cap` function is executed from ROM and hence changes in this function will not affect the policy engine behavior. The `app_update_rdo()` function should be used to update the request object in this case.

Parameters

<i>port</i>	Port index the function is performed for.
<i>src_cap</i>	Pointer to PD packet which contains source capabilities.
<i>app_resp_handler</i>	Application handler callback function.

Returns

None

6.14 app/psink.h File Reference

```
#include <pd.h>
```

Functions

- void `psnk_set_voltage` (uint8_t port, uint16_t volt_mV)
- void `psnk_set_current` (uint8_t port, uint16_t cur_10mA)
- void `psnk_enable` (uint8_t port)
- void `psnk_disable` (uint8_t port, `sink_discharge_off_cbk_t` snk_discharge_off_handler)

6.14.1 Detailed Description

Power Sink (Consumer) manager header file.

6.14.2 Function Documentation**6.14.2.1 psnk_disable()**

```
void psnk_disable (
    uint8_t port,
    sink_discharge_off_cbk_t snk_discharge_off_handler )
```

Disable the VBus power sink path and discharge VBus supply down to a safe level. This function is called by the PD stack at times when the system is not allowed to draw power from the VBus supply. The application can use this call to initiate a VBus discharge operation so that a subsequent Type-C connection is speeded up. The `snk_discharge_off_handler` callback function should be called once VBus has been discharged down to `vSafe0V`.

Parameters

<i>port</i>	Port index the function is performed for.
<i>snk_discharge_off_handler</i>	Sink Discharge fet off callback pointer

Returns

None

6.14.2.2 psnk_enable()

```
void psnk_enable (
    uint8_t port )
```

Function to enable the power consumer path to that the system can received power from the Type-C VBus. The expected voltage and maximum allowed current would have been notified through the [psnk_set_voltage\(\)](#) and [psnk_set_current\(\)](#) functions.

Parameters

<i>port</i>	Port index the function is performed for.
-------------	---

Returns

None

6.14.2.3 psnk_set_current()

```
void psnk_set_current (
    uint8_t port,
    uint16_t cur_10mA )
```

This function notifies the application code about the amount of current the system is allowed to take from the VBus power supply. The application logic should configure its load and battery charging circuits based on this value so that the power source does not see any overload condition.

Parameters

<i>port</i>	Port index the function is performed for.
<i>cur_10mA</i>	Maximum allowed current in 10mA units.

Returns

None

6.14.2.4 psnk_set_voltage()

```
void psnk_set_voltage (
    uint8_t port,
    uint16_t volt_mV )
```

This function sets the expected VBus voltage when CCG is functioning as a sink. The voltage level is used to configure the Over-Voltage Protection on the device.

Parameters

<i>port</i>	Port index the function is performed for.
<i>volt_mV</i>	Expected VBus voltage level in mV units.

Returns

None

6.15 app/psource.h File Reference

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

Macros

- `#define PPS_CF_VBUS_DECREMENT_STEP (200u)`

Functions

- void `psrc_set_voltage` (uint8_t port, uint16_t volt_mV)
- uint32_t `psrc_get_voltage` (uint8_t port)
- void `psrc_set_current` (uint8_t port, uint16_t cur_10mA)
- void `psrc_enable` (uint8_t port, pwr_ready_cbk_t pwr_ready_handler)
- void `psrc_disable` (uint8_t port, pwr_ready_cbk_t pwr_ready_handler)

6.15.1 Detailed Description

Power source (Provider) manager header file.

6.15.2 Macro Definition Documentation**6.15.2.1 PPS_CF_VBUS_DECREMENT_STEP**

```
#define PPS_CF_VBUS_DECREMENT_STEP (200u)
```

VBUS decrement value on each Current foldback event

6.15.3 Function Documentation

6.15.3.1 psrc_disable()

```
void psrc_disable (
    uint8_t port,
    pwr_ready_cbk_t pwr_ready_handler )
```

This function disables the VBus power supply. If a non-NULL `pwr_ready_handler` callback is specified, the function can return after starting the VBus disable operation. The callback will be called once the VBus voltage has been safely brought to vSafe0V. If the callback is NULL, the function is expected to return after shutting down the supply without initiating any VBus discharge sequence.

Parameters

<i>port</i>	Port index the function is performed for.
<i>pwr_ready_handler</i>	Application handler callback function.

Returns

None

6.15.3.2 psrc_enable()

```
void psrc_enable (
    uint8_t port,
    pwr_ready_cbk_t pwr_ready_handler )
```

This function enables the VBus power supply. The voltage and current to be supplied would have been specified through the [psrc_set_voltage\(\)](#) and [psrc_set_current\(\)](#) calls before the supply is enabled. The function returns as soon as the supply enable operation has been started. The `pwr_ready_handler` is expected to be called once VBus voltage has stabilized at the desired level.

Parameters

<i>port</i>	Port index the function is performed for.
<i>pwr_ready_handler</i>	Application handler callback function.

Returns

None

6.15.3.3 psrc_get_voltage()

```
uint32_t psrc_get_voltage (
    uint8_t port )
```

This function gets the VBus source voltage that is currently configured. This is different from the [vbus_get_value\(\)](#) function which measures the actual VBus voltage.

Parameters

<i>port</i>	Port index the function is performed for.
-------------	---

Returns

Voltage in mV units.

6.15.3.4 psrc_set_current()

```
void psrc_set_current (
    uint8_t port,
    uint16_t cur_10mA )
```

This function sets the VBus source current limit. The current limits are used to configure the current sensing circuits to trigger fault indication in case of overload.

Parameters

<i>port</i>	Port index the function is performed for.
<i>cur_10mA</i>	Current in 10mA units.

Returns

None

6.15.3.5 psrc_set_voltage()

```
void psrc_set_voltage (
    uint8_t port,
    uint16_t volt_mV )
```

This function sets the VBus source voltage to the desired value. It also updates the voltage thresholds associated with protection schemes such as OVP and UVP.

Parameters

<i>port</i>	Port index the function is performed for.
<i>volt_mV</i>	Voltage in mV units.

Returns

None

6.16 app/swap.h File Reference

```
#include <config.h>
#include <pd.h>
```

Functions

- void [eval_dr_swap](#) (uint8_t port, [app_resp_cbk_t](#) app_resp_handler)
- void [eval_pr_swap](#) (uint8_t port, [app_resp_cbk_t](#) app_resp_handler)

- void `eval_vconn_swap` (uint8_t port, [app_resp_cbk_t](#) app_resp_handler)
- void `eval_fr_swap` (uint8_t port, [app_resp_cbk_t](#) app_resp_handler)

6.16.1 Detailed Description

Swap request (PR_SWAP, DR_SWAP, VCONN_SWAP) handlers.

6.16.2 Function Documentation

6.16.2.1 eval_dr_swap()

```
void eval_dr_swap (
    uint8_t port,
    app\_resp\_cbk\_t app_resp_handler )
```

This function evaluates Data role swap request.

Parameters

<i>port</i>	Port index the function is performed for.
<i>app_resp_handler</i>	Application handler callback function.

Returns

None

6.16.2.2 eval_fr_swap()

```
void eval_fr_swap (
    uint8_t port,
    app\_resp\_cbk\_t app_resp_handler )
```

This function evaluates Fast role swap request.

Parameters

<i>port</i>	Port index the function is performed for.
<i>app_resp_handler</i>	Application handler callback function.

Returns

None

6.16.2.3 eval_pr_swap()

```
void eval_pr_swap (
    uint8_t port,
```



```
app_resp_cbk_t app_resp_handler )
```

This function evaluates Power role swap request.

Parameters

<i>port</i>	Port index the function is performed for.
<i>app_resp_handler</i>	Application handler callback function.

Returns

None

6.16.2.4 eval_vconn_swap()

```
void eval_vconn_swap (
    uint8_t port,
    app_resp_cbk_t app_resp_handler )
```

This function evaluates VConn swap request.

Parameters

<i>port</i>	Port index the function is performed for.
<i>app_resp_handler</i>	Application handler callback function.

Returns

None

6.17 app/vdm.h File Reference

```
#include <pd.h>
```

Functions

- void [vdm_data_init](#) (uint8_t port)
- void [vdm_update_data](#) (uint8_t port, uint8_t id_vdo_cnt, uint8_t *id_vdo_p, uint8_t svid_vdo_cnt, uint8_t *svid_vdo_p, uint16_t mode_resp_len, uint8_t *mode_resp_p)
- void [eval_vdm](#) (uint8_t port, const [pd_packet_t](#) *vdm, [vdm_resp_cbk_t](#) vdm_resp_handler)
- void [eval_enter_usb](#) (uint8_t port, const [pd_packet_t](#) *eudo, [app_resp_cbk_t](#) app_resp_handler)

6.17.1 Detailed Description

Vendor Defined Message (VDM) handler header file.

6.17.2 Function Documentation

6.17.2.1 eval_enter_usb()

```
void eval_enter_usb (
    uint8_t port,
    const pd_packet_t * eudo,
    app_resp_cbk_t app_resp_handler )
```

Function to evaluate an Enter_USB request and report whether it should be accepted or rejected.

Parameters

<i>port</i>	Port index the function is performed for.
<i>eudo</i>	Pointer to the Enter_USB packet.
<i>app_resp_handler</i>	Response callback through which the response is passed to the PD stack.

Returns

None

6.17.2.2 eval_vdm()

```
void eval_vdm (
    uint8_t port,
    const pd_packet_t * vdm,
    vdm_resp_cbk_t vdm_resp_handler )
```

This function is responsible for analysing and processing received VDM. This function also makes a decision about necessity of response to the received VDM.

Parameters

<i>port</i>	Port index the function is performed for.
<i>vdm</i>	Pointer to pd packet which contains received VDM.
<i>vdm_resp_handler</i>	VDM handler callback function.

Returns

None

6.17.2.3 vdm_data_init()

```
void vdm_data_init (
    uint8_t port )
```

Store the VDM data from the configuration table.

This function retrieves the VDM data (for CCG as UFP) that is stored in the configuration table and stores it in the run-time data structures.

Parameters

<i>port</i>	USB-PD port for which the data is to be stored.
-------------	---

Returns

None.

6.17.2.4 vdm_update_data()

```
void vdm_update_data (
    uint8_t port,
    uint8_t id_vdo_cnt,
    uint8_t * id_vdo_p,
    uint8_t svid_vdo_cnt,
    uint8_t * svid_vdo_p,
    uint16_t mode_resp_len,
    uint8_t * mode_resp_p )
```

This function allows the VDM data for CCG to be changed.

This function allows the user to change the VDM responses that CCG sends for D_ID, D_SVID and D_MODE requests. The default responses are taken from the configuration table. This function allows the user to change the response data. The caller is responsible to ensure that the responses are not changed while CCG is already in contract as a UFP.

Parameters

<i>port</i>	PD port for which responses are to be changed.
<i>id_vdo_cnt</i>	Number of VDOs in the D_ID response.
<i>id_vdo_p</i>	Pointer to the actual D_ID response in memory.
<i>svid_vdo_cnt</i>	Number of VDOs in the D_SVID response. Should be less than 8.
<i>svid_vdo_p</i>	Pointer to the actual D_SVID response in memory.
<i>mode_resp_len</i>	Total length of mode response. This includes the D_MODE responses for each supported SVID, along with the corresponding header fields.
<i>mode_resp_p</i>	Pointer to all of the mode responses in memory.

Returns

None

6.18 hpiss/hpi.h File Reference

```
#include <stdbool.h>
#include <stdint.h>
#include <config.h>
#include "status.h"
#include "i2c.h"
#include "battery_charging.h"
#include "pd.h"
#include "dpm.h"
#include "app.h"
```

Macros

- #define [HPI_ADDR_I2C_CFG_LOW](#) (0x40)

- #define [HPI_ADDR_I2C_CFG_HIGH](#) (0x42)
- #define [HPI_ADDR_I2C_CFG_FLOAT](#) (0x08)

Typedefs

- typedef [uint8_t](#)(* [hpi_write_cb_t](#)) ([uint16_t](#) reg_addr, [uint8_t](#) wr_size, [uint8_t](#) *wr_data)

Enumerations

- enum [hpi_ucsi_status_reg_t](#) { [HPI_UCSI_STATUS_STARTED](#) = 0x00, [HPI_UCSI_STATUS_CMD_IN_PROGRESS](#), [HPI_UCSI_STATUS_EVENT_PENDING](#) }
- enum [hpi_ucsi_control_cmds_t](#) { [HPI_UCSI_START_CMD](#) = 0x01, [HPI_UCSI_STOP_CMD](#), [HPI_UCSI_SILENCE_CMD](#), [HPI_UCSI_SIG_CONNECT_EVT_CMD](#) }
- enum [hpi_reg_section_t](#) { [HPI_REG_SECTION_DEV](#) = 0, [HPI_REG_SECTION_PORT_0](#), [HPI_REG_SECTION_PORT_1](#), [HPI_REG_SECTION_DEV_A](#) = 0x06, [HPI_REG_SECTION_UCSI](#) = 0x0F, [HPI_REG_SECTION_ALL](#) }
- enum [hpi_reg_part_t](#) { [HPI_REG_PART_REG](#) = 0, [HPI_REG_PART_DATA](#) = 1, [HPI_REG_PART_FLASH](#) = 2, [HPI_REG_PART_PDDATA_READ](#) = 4, [HPI_REG_PART_PDDATA_READ_H](#) = 5, [HPI_REG_PART_PDDATA_WRITE](#) = 8, [HPI_REG_PART_PDDATA_WRITE_H](#) = 9 }
- enum [hpi_boot_prio_conf_t](#) { [HPI_BOOT_PRIO_LAST_FLASHED](#) = 0, [HPI_BOOT_PRIO_APP_PRIO_ROW](#), [HPI_BOOT_PRIO_FW1](#), [HPI_BOOT_PRIO_FW2](#) }
- enum [i2c_owner_t](#) { [I2C_OWNER_UCSI](#) = 0, [I2C_OWNER_HPI](#) }

Functions

- void [hpi_init](#) ([uint8_t](#) scb_idx)
- void [hpi_deinit](#) (void)
- void [hpi_task](#) (void)
- bool [hpi_reg_enqueue_event](#) ([hpi_reg_section_t](#) section, [uint8_t](#) status, [uint16_t](#) length, [uint8_t](#) *data)
- void [hpi_pd_event_handler](#) ([uint8_t](#) port, [app_evt_t](#) evt, const void *data)
- bool [hpi_is_ec_ready](#) (void)
- void [hpi_update_versions](#) ([uint8_t](#) *bl_version, [uint8_t](#) *fw1_version, [uint8_t](#) *fw2_version)
- bool [hpi_is_accessed](#) (void)
- void [hpi_set_mode_regs](#) ([uint8_t](#) dev_mode, [uint8_t](#) mode_reason)
- void [hpi_update_fw_locations](#) ([uint16_t](#) fw1_location, [uint16_t](#) fw2_location)
- bool [hpi_is_vdm_ec_ctrl_enabled](#) ([uint8_t](#) port)
- bool [hpi_is_extd_msg_ec_ctrl_enabled](#) ([uint8_t](#) port)
- [uint8_t](#) [hpid_get_ec_active_modes](#) ([uint8_t](#) port)
- bool [hpi_sleep_allowed](#) (void)
- bool [hpi_sleep](#) (void)
- [uint8_t](#) [hpi_get_port_enable](#) (void)
- void [hpi_set_no_boot_mode](#) (bool enable)
- void [hpi_set_fixed_slave_address](#) ([uint8_t](#) slave_addr)
- void [hpi_set_ec_interrupt](#) (bool enable)
- void [hpi_send_fw_ready_event](#) (void)
- void [hpi_set_flash_params](#) ([uint32_t](#) flash_size, [uint16_t](#) row_size, [uint16_t](#) row_cnt, [uint16_t](#) bl_last_row)
- [ccg_status_t](#) [hpi_init_userdef_regs](#) ([uint16_t](#) reg_addr, [uint8_t](#) size, [uint8_t](#) *data)
- void [hpi_set_userdef_write_handler](#) ([hpi_write_cb_t](#) wr_handler)
- void [hpi_set_port_event_mask](#) ([uint8_t](#) port, [uint32_t](#) mask)
- void [hpi_send_hw_error_event](#) ([uint8_t](#) port, [uint8_t](#) err_type)
- [uint8_t](#) [hpi_get_sys_pwr_state](#) (void)

- void `hpi_set_boot_priority_conf` (uint8_t conf)
- void `hpi_update_pdo_change` (bool disable)
- void `hpi_set_event` (uint8_t evt_code)
- void `hpi_clear_event` (uint8_t evt_code)
- void `ucsi_reg_reset` (void)
- void `ucsi_clear_status_bit` (uint8_t bit_idx)
- void `ucsi_set_status_bit` (uint8_t bit_idx)
- uint8_t `hpi_get_ucsi_control` (void)
- uint8_t `ucsi_get_status_bit` (uint8_t bit_idx)
- void `hpi_set_hpi_version` (uint32_t hpi_vers)

6.18.1 Detailed Description

Host Processor Interface (HPI) header file.

6.18.2 Macro Definition Documentation

6.18.2.1 HPI_ADDR_I2C_CFG_FLOAT

```
#define HPI_ADDR_I2C_CFG_FLOAT (0x08)
```

I2C slave address to be used for HPI interface, when the I2C_CFG pin is sensed as FLOATING.

6.18.2.2 HPI_ADDR_I2C_CFG_HIGH

```
#define HPI_ADDR_I2C_CFG_HIGH (0x42)
```

I2C slave address to be used for HPI interface, when the I2C_CFG pin is sensed as HIGH.

6.18.2.3 HPI_ADDR_I2C_CFG_LOW

```
#define HPI_ADDR_I2C_CFG_LOW (0x40)
```

I2C slave address to be used for HPI interface, when the I2C_CFG pin is sensed as LOW.

6.18.3 Typedef Documentation

6.18.3.1 hpi_write_cb_t

```
hpi_write_cb_t
```

Handler for HPI register writes.

Returns

Type of response to be sent to the EC. Only a single byte response can be sent from here. Use `hpi_reg_enqueue_event` to send longer responses.

6.18.4 Enumeration Type Documentation

6.18.4.1 hpi_boot_prio_conf_t

enum `hpi_boot_prio_conf_t`

Enumeration showing possible boot priority configurations for the firmware application.

Enumerator

HPI_BOOT_PRIO_LAST_FLASHED	Last flashed firmware is prioritized.
HPI_BOOT_PRIO_APP_PRIO_ROW	Priority defined used App Priority flash row.
HPI_BOOT_PRIO_FW1	FW1 is always prioritized.
HPI_BOOT_PRIO_FW2	FW2 is always prioritized.

6.18.4.2 hpi_reg_part_t

enum `hpi_reg_part_t`

Types of HPI register/memory regions.

Enumerator

HPI_REG_PART_REG	Register region.
HPI_REG_PART_DATA	Data memory for device section.
HPI_REG_PART_FLASH	Flash memory.
HPI_REG_PART_PDDATA_READ	Read Data memory for port section.
HPI_REG_PART_PDDATA_READ_H	Upper fraction of read data memory for port section.
HPI_REG_PART_PDDATA_WRITE	Write Data memory for port section.
HPI_REG_PART_PDDATA_WRITE_H	Upper fraction of write data memory for port section.

6.18.4.3 hpi_reg_section_t

enum `hpi_reg_section_t`

HPI register section definitions.

HPI registers are grouped into sections corresponding to the functions that are supported.

Enumerator

HPI_REG_SECTION_DEV	Device information registers.
HPI_REG_SECTION_PORT_0	USB-PD Port 0 related registers.
HPI_REG_SECTION_PORT_1	USB-PD Port 1 related registers.
HPI_REG_SECTION_DEV_AUTO	HPI Auto related registers
HPI_REG_SECTION_UCSI	UCSI related registers.
HPI_REG_SECTION_ALL	Special definition to select all register spaces.

6.18.4.4 hpi_ucsi_control_cmds_t

enum `hpi_ucsi_control_cmds_t`

HPI UCSI Control commands.

Commands to control the UCSI interface.

Enumerator

HPI_UCSI_START_CMD	UCSI Control Register Start value. This commands starts the UCSI Interface.
HPI_UCSI_STOP_CMD	UCSI Control Register Stop value. This command stops the UCSI interface.
HPI_UCSI_SILENCE_CMD	UCSI Control Register Silence value. This command silences the UCSI Port.
HPI_UCSI_SIG_CONNECT_EVT_CMD	UCSI Control Register Signal Connect Event value. EC sends this command to ask CCG to send connect event information to OS.

6.18.4.5 hpi_ucsi_status_reg_t

enum `hpi_ucsi_status_reg_t`

HPI UCSI Status register values.

Status values for the UCSI Status register defined in the HPI register space.

Enumerator

HPI_UCSI_STATUS_STARTED	Status value to indicate UCSI is started.
HPI_UCSI_STATUS_CMD_IN_PROGRESS	Status value to indicate UCSI Command in progress.
HPI_UCSI_STATUS_EVENT_PENDING	Status value to indicate UCSI event pending.

6.18.4.6 i2c_owner_t

enum `i2c_owner_t`

List of possible owners for the I2C slave interface. The interface can be used for one of HPI or UCSI functionality at a time.

Enumerator

I2C_OWNER_UCSI	I2C interface used for UCSI commands.
I2C_OWNER_HPI	I2C interface used for HPI commands.

6.18.5 Function Documentation

6.18.5.1 hpi_clear_event()

```
void hpi_clear_event (
    uint8_t evt_code )
```

Clear the Interrupt Status register and de-assert the EC_INT pin.

Parameters

<i>evt_code</i>	Event code to be cleared.
-----------------	---------------------------

Returns

None

6.18.5.2 hpi_deinit()

```
void hpi_deinit (
    void )
```

De-initialize the HPI interface.

This function can be used to de-initialize the HPI interface. This can be used in applications where the HPI master (Billboard controller) in the system may be powered off under control of the CCG device.

Returns

None

6.18.5.3 hpi_get_port_enable()

```
uint8_t hpi_get_port_enable (
    void )
```

Get the Port Enable register value.

Returns

The Port Enable HPI register value.

6.18.5.4 hpi_get_sys_pwr_state()

```
uint8_t hpi_get_sys_pwr_state (
    void )
```

Get the content of the HPI system power state register.

Returns

The 8-bit unsigned content of the HPI syspwr_state register.

6.18.5.5 hpi_get_ucsi_control()

```
uint8_t hpi_get_ucsi_control (
    void )
```

Get the UCSI Control register value.

Returns

The UCSI Control HPI register value.

6.18.5.6 hpi_init()

```
void hpi_init (
    uint8_t scb_idx )
```

Initialize the HPI interface.

This function initializes the I2C interface and EC_INT GPIO used for HPI hardware interface, and initializes all HPI registers to their default values.

Parameters

<i>scb_idx</i>	Index of SCB block to be used for HPI. Please note that this parameter is not validated, and it is the caller's responsibility to pass the correct value.
----------------	---

Returns

None

6.18.5.7 hpi_init_userdef_regs()

```
ccg_status_t hpi_init_userdef_regs (
    uint16_t reg_addr,
    uint8_t size,
    uint8_t * data )
```

This function initializes the user-defined HPI registers.

This function is used to initialize the contents of the user-defined registers that are part of the HPI register space.

Parameters

<i>reg_addr</i>	The base address of registers to be updated. Should be in the user defined address region.
<i>size</i>	Number of registers to be updated. The upper limit should not exceed the user defined address region.
<i>data</i>	Buffer containing data to be copied into HPI registers.

Returns

CCG_STAT_SUCCESS if operation is successful, CCG_STAT_BAD_PARAM otherwise.

6.18.5.8 hpi_is_accessed()

```
bool hpi_is_accessed (
    void )
```

Check whether any HPI accesses have happened since start-up.

Returns

True if any HPI read/write access has happened.

6.18.5.9 hpi_is_ec_ready()

```
bool hpi_is_ec_ready (
    void )
```

Check whether EC init complete event has been received.

This function is used by the application to check whether the EC has sent the EC initialization complete event notification.

Returns

true if EC init has been received, false otherwise.

6.18.5.10 hpi_is_extd_msg_ec_ctrl_enabled()

```
bool hpi_is_extd_msg_ec_ctrl_enabled (
    uint8_t port )
```

Check whether handling of extended messages by EC is enabled. If not enabled, CCG firmware will automatically respond with NOT_SUPPORTED messages.

Parameters

<i>port</i>	USB-PD port to check the configuration for.
-------------	---

Returns

true if EC handling of extended messages is enabled, false otherwise.

6.18.5.11 hpi_is_vdm_ec_ctrl_enabled()

```
bool hpi_is_vdm_ec_ctrl_enabled (
    uint8_t port )
```

Check whether EC control of VDMs is enabled.

Parameters

<i>port</i>	USB-PD port to check the configuration for.
-------------	---

Returns

true if EC control is enabled, false otherwise.

6.18.5.12 hpi_pd_event_handler()

```
void hpi_pd_event_handler (
    uint8_t port,
    app_evt_t evt,
    const void * data )
```

Handler for PD events reported from the stack.

Internal function used to receive PD events from the stack and to update the HPI registers.

Parameters

<i>port</i>	PD port corresponding to the event.
<i>evt</i>	Event that is being notified.
<i>data</i>	Data associated with the event. This is an opaque pointer that needs to be de-referenced based on event type.

Returns

None

6.18.5.13 hpi_reg_enqueue_event()

```
bool hpi_reg_enqueue_event (
    hpi_reg_section_t section,
    uint8_t status,
    uint16_t length,
    uint8_t * data )
```

Enqueue an event to the EC through the HPI interface.

This function is used by the PD stack and application layers to send event notifications to the EC through the HPI registers.

Please note that only responses without associated data can be sent through the response register for the HPI_↔REG_SECTION_DEV section. If any additional response data is required, please use the user defined registers or the response register associated with HPI_REG_SECTION_PORT_0 or HPI_REG_SECTION_PORT_1.

Parameters

<i>section</i>	Register section through which event is to be reported.
<i>status</i>	The event code to be stored into the response register.
<i>length</i>	Length of the data associated with the event.
<i>data</i>	Pointer to buffer containing data associated with the event.

Returns

true if the event queue has space for the event, false if there is an overflow.

6.18.5.14 hpi_send_fw_ready_event()

```
void hpi_send_fw_ready_event (
    void )
```

Send a FW ready notification through HPI to the EC. This event is sent to the EC to indicate that the device is out of reset and has loaded firmware.

Returns

None

6.18.5.15 hpi_send_hw_error_event()

```
void hpi_send_hw_error_event (
    uint8_t port,
    uint8_t err_type )
```

Notify EC about a system hardware access error.

Parameters

<i>port</i>	PD port index.
<i>err_type</i>	Type of error detected.

Returns

None

6.18.5.16 hpi_set_boot_priority_conf()

```
void hpi_set_boot_priority_conf (
    uint8_t conf )
```

Update the firmware boot priority configuration reported through HPI.

Parameters

<i>conf</i>	Firmware boot priority supported by the firmware.
-------------	---

Returns

None

6.18.5.17 hpi_set_ec_interrupt()

```
void hpi_set_ec_interrupt (
    bool enable )
```

Configure HPI to use EC_INT pin for interrupt mode.

Parameters

<i>enable</i>	Whether to enable or disable interrupt mode.
---------------	--

Returns

None

6.18.5.18 hpi_set_event()

```
void hpi_set_event (
    uint8_t evt_code )
```

Update the Interrupt Status register and assert the EC_INT pin.

Parameters

<i>evt_code</i>	Event code to be set.
-----------------	-----------------------

Returns

None

6.18.5.19 hpi_set_fixed_slave_address()

```
void hpi_set_fixed_slave_address (
    uint8_t slave_addr )
```

Set the I2C slave address to be used for the HPI interface.

Parameters

<i>slave_addr</i>	Slave address to be used.
-------------------	---------------------------

Returns

None

6.18.5.20 hpi_set_flash_params()

```
void hpi_set_flash_params (
    uint32_t flash_size,
```

```
uint16_t row_size,
uint16_t row_cnt,
uint16_t bl_last_row )
```

Set the CCG device flash parameters. These values are used for the device status reporting and firmware update implementation.

Parameters

<i>flash_size</i>	Total device flash size in bytes.
<i>row_size</i>	Size of each flash row in bytes.
<i>row_cnt</i>	Number of flash rows on the device.
<i>bl_last_row</i>	Last flash row assigned to boot-loader.

Returns

None

6.18.5.21 hpi_set_hpi_version()

```
void hpi_set_hpi_version (
    uint32_t hpi_vers )
```

This function sets the hpi version info.

Parameters

<i>hpi_vers</i>	hpi version informaraion.
-----------------	---------------------------

Returns

void.

6.18.5.22 hpi_set_mode_regs()

```
void hpi_set_mode_regs (
    uint8_t dev_mode,
    uint8_t mode_reason )
```

Set device mode and reason register values.

This is an internal function used to update the device mode and boot mode reason HPI registers.

Parameters

<i>dev_mode</i>	Value to be set into the device mode register.
<i>mode_reason</i>	Value to be set into the boot mode reason register.

Returns

void

6.18.5.23 hpi_set_no_boot_mode()

```
void hpi_set_no_boot_mode (
    bool enable )
```

Configure HPI to operate in No-boot support mode.

Parameters

<i>enable</i>	Whether to enable no-boot mode.
---------------	---------------------------------

Returns

None

6.18.5.24 hpi_set_port_event_mask()

```
void hpi_set_port_event_mask (
    uint8_t port,
    uint32_t mask )
```

Update the event mask value for the specified PD port.

Parameters

<i>port</i>	PD port index.
<i>mask</i>	Event mask value to be set.

Returns

None

6.18.5.25 hpi_set_userdef_write_handler()

```
void hpi_set_userdef_write_handler (
    hpi_write_cb_t wr_handler )
```

Enable handling of user-defined register writes in the Application.

This function is used to enable handling of EC writes to the user-defined HPI register region in the application code.

Parameters

<i>wr_handler</i>	Pointer to function that handles the received HPI writes.
-------------------	---

6.18.5.26 hpi_sleep()

```
bool hpi_sleep (
    void )
```

Prepare the HPI interface for device deep sleep.

This function checks whether the I2C interface is idle so that the CCG device can enter deep sleep mode. It also enables an I2C address match as a wake-up trigger from deep sleep. `hpi_sleep_allowed` should have been called prior to calling this function.

Returns

true if the HPI interface is ready for sleep, false otherwise.

6.18.5.27 hpi_sleep_allowed()

```
bool hpi_sleep_allowed (
    void )
```

Check if the CCG device can be put into deep-sleep.

Returns

true if deep sleep is possible, false otherwise.

6.18.5.28 hpi_task()

```
void hpi_task (
    void )
```

HPI task handler.

This function handles the commands from the EC through the HPI registers. HPI writes from the EC are handled in interrupt context, and any associated work is queued to be handled by this function. The `hpi_task` is expected to be called periodically from the main task loop of the firmware application.

Returns

None

6.18.5.29 hpi_update_fw_locations()

```
void hpi_update_fw_locations (
    uint16_t fw1_location,
    uint16_t fw2_location )
```

Update the firmware location HPI registers.

This is an internal function used to update the firmware binary location HPI registers.

Parameters

<i>fw1_location</i>	Flash row where FW1 is located.
<i>fw2_location</i>	Flash row where FW2 is located.

Returns

void

6.18.5.30 hpi_update_pdo_change()

```
void hpi_update_pdo_change (
    bool disable )
```

Enable/disable PDO update through HPI.

Parameters

<i>disable</i>	Whether PDO update is to be disabled.
----------------	---------------------------------------

Returns

None

6.18.5.31 hpi_update_versions()

```
void hpi_update_versions (
    uint8_t * bl_version,
    uint8_t * fw1_version,
    uint8_t * fw2_version )
```

Update firmware version information in HPI registers.

This is an internal function used to update the firmware version information in the HPI registers.

Parameters

<i>bl_version</i>	Buffer containing Bootloader version information.
<i>fw1_version</i>	Buffer containing firmware-1 version information.
<i>fw2_version</i>	Buffer containing firmware-2 version information.

Returns

void

6.18.5.32 hpid_get_ec_active_modes()

```
uint8_t hpid_get_ec_active_modes (
    uint8_t port )
```

Get the active EC alternate modes value.

Parameters

<i>port</i>	USB-PD to check the configuration for.
-------------	--

Returns

The Active EC modes setting programmed by EC.

6.18.5.33 ucsi_clear_status_bit()

```
void ucsi_clear_status_bit (
    uint8_t bit_idx )
```

Clear appropriate bit_idx to 0 in the status register.

Returns

None

6.18.5.34 ucsi_get_status_bit()

```
uint8_t ucsi_get_status_bit (
    uint8_t bit_idx )
```

Get the UCSI Status bit.

Returns

The UCSI Status bit value.

6.18.5.35 ucsi_reg_reset()

```
void ucsi_reg_reset (
    void )
```

Clear status & control register.

Returns

None

6.18.5.36 ucsi_set_status_bit()

```
void ucsi_set_status_bit (
    uint8_t bit_idx )
```

Set appropriate bit_idx to 1 in the status register.

Returns

None

6.19 pd_common/dpm.h File Reference

```
#include "pd.h"
#include "status.h"
```

Functions

- [ccg_status_t dpm_init](#) (uint8_t port, [app_cbk_t](#) *app_cbk)
- [ccg_status_t dpm_start](#) (uint8_t port)
- [ccg_status_t dpm_stop](#) (uint8_t port)
- [ccg_status_t dpm_disable](#) (uint8_t port)
- [bool dpm_deepsleep](#) (void)
- [bool dpm_wakeup](#) (void)
- [bool dpm_sleep](#) (void)
- [ccg_status_t dpm_task](#) (uint8_t port)
- [const dpm_status_t * dpm_get_info](#) (uint8_t port)
- [void dpm_update_def_cable_cap](#) (uint16_t def_cur)
- [uint16_t dpm_get_def_cable_cap](#) (void)
- [void dpm_update_snk_wait_cap_period](#) (uint16_t period)
- [uint16_t dpm_get_snk_wait_cap_period](#) (void)
- [void dpm_update_mux_enable_wait_period](#) (uint16_t period)
- [uint16_t dpm_get_mux_enable_wait_period](#) (void)
- [ccg_status_t dpm_pd_command](#) (uint8_t port, [dpm_pd_cmd_t](#) cmd, [dpm_pd_cmd_buf_t](#) *buf_ptr, [dpm_pd_cmd_cbk_t](#) cmd_cbk)
- [ccg_status_t dpm_pd_command_ec](#) (uint8_t port, [dpm_pd_cmd_t](#) cmd, [dpm_pd_cmd_buf_t](#) *buf_ptr, [dpm_pd_cmd_cbk_t](#) cmd_cbk)
- [ccg_status_t dpm_typec_command](#) (uint8_t port, [dpm_typec_cmd_t](#) cmd, [dpm_typec_cmd_cbk_t](#) cmd_cbk)
- [ccg_status_t dpm_update_swap_response](#) (uint8_t port, uint8_t value)
- [ccg_status_t dpm_update_src_cap](#) (uint8_t port, uint8_t count, [pd_do_t](#) *pdo)
- [ccg_status_t dpm_update_src_cap_mask](#) (uint8_t port, uint8_t mask)
- [ccg_status_t dpm_update_snk_cap](#) (uint8_t port, uint8_t count, [pd_do_t](#) *pdo)
- [ccg_status_t dpm_update_snk_cap_mask](#) (uint8_t port, uint8_t mask)
- [ccg_status_t dpm_update_snk_max_min](#) (uint8_t port, uint8_t count, uint16_t *max_min)
- [ccg_status_t dpm_update_port_config](#) (uint8_t port, uint8_t role, uint8_t dflt_role, uint8_t toggle_en, uint8_t try_src_snk_en)
- [ccg_status_t dpm_is_rdo_valid](#) (uint8_t port, [pd_do_t](#) rdo)
- [uint8_t dpm_get_polarity](#) (uint8_t port)
- [ccg_status_t dpm_typec_deassert_rp_rd](#) (uint8_t port, uint8_t channel)
- [void dpm_update_port_status](#) (uint8_t port, uint8_t *status_p, uint8_t offset, uint8_t byte_cnt)
- [uint32_t dpm_get_pd_port_status](#) (uint8_t port)
- [ccg_status_t dpm_downgrade_pd_port_rev](#) (uint8_t port)
- [void dpm_update_frs_enable](#) (uint8_t port, bool frs_rx_en, bool frs_tx_en)
- [void dpm_update_ext_src_cap](#) (uint8_t port, uint8_t *buf_p)
- [void dpm_update_ext_snk_cap](#) (uint8_t port, uint8_t *buf_p)
- [ccg_status_t dpm_prot_reset](#) (uint8_t port, [sop_t](#) sop)
- [ccg_status_t dpm_prot_reset_rx](#) (uint8_t port, [sop_t](#) sop)
- [ccg_status_t dpm_pe_stop](#) (uint8_t port)
- [ccg_status_t dpm_set_alert](#) (uint8_t port, [pd_do_t](#) alert_ado)
- [ccg_status_t dpm_set_cf](#) (uint8_t port, bool status)
- [ccg_status_t dpm_clear_hard_reset_count](#) (uint8_t port)
- [ccg_status_t dpm_set_fault_active](#) (uint8_t port)
- [ccg_status_t dpm_clear_fault_active](#) (uint8_t port)
- [ccg_status_t dpm_set_chunk_transfer_running](#) (uint8_t port, [pd_ams_type](#) ams_type)

- [ccg_status_t dpm_send_hard_reset](#) (uint8_t port, uint8_t reason)
- [int8_t dpm_get_sink_detach_margin](#) (uint8_t port)
- [uint16_t dpm_get_sink_detach_voltage](#) (uint8_t port)
- [void dpm_pps_task](#) (uint8_t port)
- [void dpm_update_ndiscover_identity_count](#) (uint8_t count)
- [uint8_t dpm_get_ndiscover_identity_count](#) (void)
- [pd_stack_conf_t dpm_get_stack_config](#) (void)
- [bool dpm_refresh_src_cap](#) (uint8_t port)
- [bool dpm_refresh_snk_cap](#) (uint8_t port)
- [void dpm_set_delay_src_cap_start](#) (uint8_t port, uint16_t delay)
- [void dpm_update_rp_audio_accessory](#) (rp_term_t rp_sel)
- [rp_term_t dpm_get_rp_audio_accessory](#) (void)
- [usb_data_sig_t dpm_get_cable_usb_cap](#) (uint8_t port)
- [uint16_t dpm_get_vbus_voltage](#) (uint8_t port)
- [ccg_status_t dpm_pd3_src_rp_flow_control](#) (uint8_t port, bool block_sink_ams)
- [bool dpm_is_idle](#) (void)
- [bool dpm_is_accessory_mode_disabled](#) (uint8_t port)
- [void dpm_set_accessory_mode_disabled](#) (uint8_t port, bool is_disabled)
- [bool dpm_is_rp_detach_detect_disabled](#) (uint8_t port)
- [void dpm_set_rp_detach_detect_disabled](#) (uint8_t port, bool is_disabled)

Variables

- [port_type_t gl_dpm_port_type](#) [NO_OF_TYPEC_PORTS]

6.19.1 Detailed Description

Device Policy Manager (DPM) header file.

6.19.2 Function Documentation

6.19.2.1 dpm_clear_fault_active()

```
ccg_status_t dpm_clear_fault_active (
    uint8_t port )
```

This function clears the internal fault active flags.

Parameters

<i>port</i>	Port index
-------------	------------

Returns

ccg_status_t

6.19.2.2 dpm_clear_hard_reset_count()

```
ccg_status_t dpm_clear_hard_reset_count (
```

```
uint8_t port )
```

This function clears hard reset count. This is specifically provided for VBUS fault handlers.

Parameters

<i>port</i>	Port index
-------------	------------

Returns

ccg_status_t

6.19.2.3 dpm_deepsleep()

```
bool dpm_deepsleep (
    void )
```

This function configures the device policy manager, for all ports, so that the system can go to deepsleep. This function does not put the system into deepsleep but only performs the necessary tasks to enter deepsleep, if entry is possible.

Returns

Returns true if deepsleep is possible and configured, false otherwise.

6.19.2.4 dpm_disable()

```
ccg_status_t dpm_disable (
    uint8_t port )
```

This function disables PD port operation and limits it to receiving hard reset signaling.

Parameters

<i>port</i>	Port index.
-------------	-------------

Returns

CCG_STAT_SUCCESS if operation is successful, CCG_STAT_BAD_PARAM otherwise.

6.19.2.5 dpm_downgrade_pd_port_rev()

```
ccg_status_t dpm_downgrade_pd_port_rev (
    uint8_t port )
```

Downgrade the PD port revision from 3.0 to 2.0.

Parameters

<i>port</i>	PD port to be updated.
-------------	------------------------

Returns

CCG_STAT_SUCCESS if operation is successful, CCG_STAT_FAILURE otherwise.

6.19.2.6 dpm_get_cable_usb_cap()

```
usb_data_sig_t dpm_get_cable_usb_cap (
    uint8_t port )
```

Function to retrieve the type of USB signalling supported by the Type-C cable in use. The information is calculated based on the cable VDO responses obtained from the cable marker.

Parameters

<i>port</i>	PD port index.
-------------	----------------

Returns

USB data signalling supported by the cable marker if known, USB_SIG_UNKNOWN otherwise.

6.19.2.7 dpm_get_def_cable_cap()

```
uint16_t dpm_get_def_cable_cap (
    void )
```

This function returns the default cable current characteristics for the stack. The parameter is used by the stack to determine the maximum current setting allowed when no e-marked cable is present.

Returns

Default current setting in 10mA unit (3A is represented as 300).

6.19.2.8 dpm_get_info()

```
const dpm_status_t* dpm_get_info (
    uint8_t port )
```

This function returns device policy manager status information for the specified port.

Parameters

<i>port</i>	Port index.
-------------	-------------

Returns

Pointer to a structure containing the DPM status information.

Warning

The information provided by this API must not be altered by the application.

6.19.2.9 dpm_get_mux_enable_wait_period()

```
uint16_t dpm_get_mux_enable_wait_period (  
    void )
```

Returns the current delay between the MUX enable and VBus turn ON.

Returns

Current delay in ms.

6.19.2.10 dpm_get_ndiscover_identity_count()

```
uint8_t dpm_get_ndiscover_identity_count (  
    void )
```

Function to retrieve the active value of maximum number of EMCA discover identity messages that should be sent.

Returns

Active count of maximum number of discover identity messages.

6.19.2.11 dpm_get_pd_port_status()

```
uint32_t dpm_get_pd_port_status (  
    uint8_t port )
```

Get the PD port status.

Parameters

<i>port</i>	PD port to be queried.
-------------	------------------------

Returns

32-bit PD port status value to be reported through HPI.

6.19.2.12 dpm_get_polarity()

```
uint8_t dpm_get_polarity (  
    uint8_t port )
```

Get the CC polarity of the Type-C connection.

Parameters

<i>port</i>	Port index.
-------------	-------------

Returns

Returns 0 if CC1 is used, and 1 if CC2 is used.

6.19.2.13 dpm_get_rp_audio_accessory()

```
rp_term_t dpm_get_rp_audio_accessory (  
    void )
```

Function to retrieve the Rp level that CCGx as source will use when connected to an audio accessory.

Returns

Rp setting that will be used when connected to audio accessory.

6.19.2.14 dpm_get_sink_detach_margin()

```
int8_t dpm_get_sink_detach_margin (  
    uint8_t port )
```

This function returns the margin on the VBus below which sink can assume detach has happened.

Parameters

<i>port</i>	Port index.
-------------	-------------

Returns

Percentage by which VBus needs to drop below nominal value for detach detection.

6.19.2.15 dpm_get_sink_detach_voltage()

```
uint16_t dpm_get_sink_detach_voltage (  
    uint8_t port )
```

This function returns the nominal VBus voltage threshold based on which sink will detect a disconnect. The sink margin returned by `dpm_get_sink_detach_margin` will be applied to this value to determine the actual detach threshold.

Parameters

<i>port</i>	Port index.
-------------	-------------

Returns

Nominal VBus voltage in mV units

6.19.2.16 dpm_get_snk_wait_cap_period()

```
uint16_t dpm_get_snk_wait_cap_period (
    void )
```

Returns the current tTypeCSnkWaitCap value used by the CCG PD stack. This function is used by the stack to get the default or user selected timer period.

Returns

Current tTypeCSnkWaitCap timer period.

6.19.2.17 dpm_get_stack_config()

```
pd_stack_conf_t dpm_get_stack_config (
    void )
```

Function to retrieve the configurable switch values at runtime.

Returns

Structure indicating current stack configuration.

6.19.2.18 dpm_get_vbus_voltage()

```
uint16_t dpm_get_vbus_voltage (
    uint8_t port )
```

Function which uses the application provided callback to measure the current VBus voltage.

Parameters

<i>port</i>	PD port index.
-------------	----------------

Returns

VBus voltage in mV units.

6.19.2.19 dpm_init()

```
cpg_status_t dpm_init (
    uint8_t port,
    app_cbk_t * app_cbk )
```

This function initializes the device policy manager with callback pointers and loads the system info from config table. This function also initializes policy engine and type c manager. This function must be called once on system init.

Parameters

<i>port</i>	Port index.
<i>app_cbk</i>	Application callback function pointer.

Returns

CCG_STAT_SUCCESS if operation is successful, CCG_STAT_BAD_PARAM otherwise.

6.19.2.20 `dpm_is_accessory_mode_disabled()`

```
bool dpm_is_accessory_mode_disabled (
    uint8_t port )
```

Function to check whether Audio/Debug accessory support is disabled for the Type-C port.

Parameters

<i>port</i>	Type-C port index.
-------------	--------------------

Returns

true if accessory mode should not be supported.

6.19.2.21 `dpm_is_idle()`

```
bool dpm_is_idle (
    void )
```

Function to check whether the Type-C/PD stack is idle.

Returns

true if stack is idle, false if any operations are pending on any port.

6.19.2.22 `dpm_is_rdo_valid()`

```
ccg_status_t dpm_is_rdo_valid (
    uint8_t port,
    pd_do_t rdo )
```

This generic function is provided by the device policy manager to evaluate any RDO with respect to current source cap of the specified port.

Parameters

<i>port</i>	Port index.
<i>rdo</i>	Request data object.

Returns**Returns**

- CCG_STAT_SUCCESS if RDO is valid
- CCG_STAT_BAD_PARAM if port index is not correct
- CCG_STAT_FAILURE if rdo is not valid.

6.19.2.23 dpm_is_rp_detach_detect_disabled()

```
bool dpm_is_rp_detach_detect_disabled (
    uint8_t port )
```

Function to check whether Rp based source detach detection is disabled for the Type-C port.

Parameters

<i>port</i>	Type-C port index.
-------------	--------------------

Returns

true if Rp based detach detection is disabled.

6.19.2.24 dpm_pd3_src_rp_flow_control()

```
ccg_status_t dpm_pd3_src_rp_flow_control (
    uint8_t port,
    bool block_sink_ams )
```

Function to specify whether we should keep the Rp termination at SinkTxNG or SinkTxOK while acting as a PD 3.0 source.

Parameters

<i>port</i>	PD port index.
<i>block_sink_ams</i>	If true, Rp will be changed to SinkTxNG; if false, Rp will be changed to SinkTxOK.

Returns

CCG_STAT_SUCCESS if DUT is a PD 3.0 source, CCG_STAT_CMD_FAILURE if DUT is not a PD 3.0 source.

6.19.2.25 dpm_pd_command()

```
ccg_status_t dpm_pd_command (
    uint8_t port,
    dpm_pd_cmd_t cmd,
    dpm_pd_cmd_buf_t * buf_ptr,
    dpm_pd_cmd_cbk_t cmd_cbk )
```

This function provides an interface for the application module to send PD commands.

Parameters

<i>port</i>	Port index.
<i>cmd</i>	Command name.
<i>buf_ptr</i>	Pointer to the command buffer.
<i>cmd_cbk</i>	Pointer to the callback function.

Returns

Returns

- CCG_STAT_SUCCESS if the command is registered
- CCG_STAT_CMD_FAILURE if the PD port is not ready for a command
- CCG_STAT_BUSY if there is another pending command.

Warning

Data received via the callback should be copied out by the application, because the buffer will be reused by the stack to store data on new message reception.

6.19.2.26 dpm_pd_command_ec()

```
ccg_status_t dpm_pd_command_ec (
    uint8_t port,
    dpm_pd_cmd_t cmd,
    dpm_pd_cmd_buf_t * buf_ptr,
    dpm_pd_cmd_cbk_t cmd_cbk )
```

This function provides an interface for the HPI module to send PD commands. This is only meant for HPI module wherein responses come from EC.

Parameters

<i>port</i>	Port index.
<i>cmd</i>	Command name.
<i>buf_ptr</i>	Pointer to the command buffer.
<i>cmd_cbk</i>	Pointer to the callback function.

Returns

Returns

- CCG_STAT_SUCCESS if the command is registered
- CCG_STAT_CMD_FAILURE if the PD port is not ready for a command
- CCG_STAT_BUSY if there is another pending command.

Warning

Data received via the callback should be copied out by the application, because the buffer will be reused by the stack to store data on new message reception.

6.19.2.27 dpm_pe_stop()

```
ccg_status_t dpm_pe_stop (
    uint8_t port )
```

This function stops the policy engine. Used in fault scenario wherein PD protocol need to be stopped but type c manager still runs.

Parameters

<i>port</i>	port index
-------------	------------

Returns

ccg_status_t

6.19.2.28 dpm_pps_task()

```
void dpm_pps_task (
    uint8_t port )
```

This function monitors the PPS activity. This function needs to be called periodically when in PPS mode of operation. Since the VBUS voltage is expected to have stabilized when invoking this function, this is not handled internally and is expected to be triggered from the application layer. The recommended periodicity for this is 100ms.

Parameters

<i>port</i>	Port index.
-------------	-------------

Returns

None.

6.19.2.29 dpm_prot_reset()

```
ccg_status_t dpm_prot_reset (
    uint8_t port,
    sop_t sop )
```

This function resets protocol layer(RX and TX) counter for a specific sop type.

Parameters

<i>port</i>	port index
<i>sop</i>	sop type

Returns

ccg_status_t

6.19.2.30 dpm_prot_reset_rx()

```
ccg_status_t dpm_prot_reset_rx (
    uint8_t port,
    sop_t sop )
```

This function resets protocol layer RX only counter for a specific sop type.

Parameters

<i>port</i>	port index
<i>sop</i>	sop type

Returns

ccg_status_t

6.19.2.31 dpm_refresh_snk_cap()

```
bool dpm_refresh_snk_cap (
    uint8_t port )
```

Function to update the Sink Capabilities based on active PD revision. This causes PDOs that are not applicable to be disabled.

Parameters

<i>port</i>	Port index.
-------------	-------------

Returns

True on successful operation.

6.19.2.32 dpm_refresh_src_cap()

```
bool dpm_refresh_src_cap (
    uint8_t port )
```

Function to update the Source Capabilities based on active PD revision and cable properties. This causes PDOs that are not applicable to be disabled.

Parameters

<i>port</i>	Port index.
-------------	-------------

Returns

True on successful operation.

6.19.2.33 dpm_send_hard_reset()

```
ccg_status_t dpm_send_hard_reset (
    uint8_t port,
    uint8_t reason )
```

Try to send a HardReset to the port partner.

Parameters

<i>port</i>	PD port index.
<i>reason</i>	Reason for the hard reset. Used for internal status tracking only.

Returns

ccg_status_t

6.19.2.34 dpm_set_accessory_mode_disabled()

```
void dpm_set_accessory_mode_disabled (
    uint8_t port,
    bool is_disabled )
```

Function to update the Audio/Debug accessory support for the Type-C port.

Parameters

<i>port</i>	Type-C port index.
<i>is_disabled</i>	Whether Audio/Debug accessory support should be disabled.

6.19.2.35 dpm_set_alert()

```
ccg_status_t dpm_set_alert (
    uint8_t port,
    pd_do_t alert_ado )
```

This function sets alert ADO on ocp/ovp fault. Stack will automatically send alert after explicit contract when alert ADO is non zero. Once alert is sent or detach happens; alert ADO will be cleared automatically by stack.

Parameters

<i>port</i>	port index
<i>alert_ado</i>	Alert Augmented Data Object.

Returns

ccg_status_t

6.19.2.36 dpm_set_cf()

```
ccg_status_t dpm_set_cf (
    uint8_t port,
    bool status )
```

This function sets/clears current foldback status. When in current foldback mode this function can be used to convey the status to stack.

Parameters

<i>port</i>	Port index
<i>status</i>	CF status

Returns

ccg_status_t

6.19.2.37 dpm_set_chunk_transfer_running()

```
ccg_status_t dpm_set_chunk_transfer_running (
    uint8_t port,
    pd_ams_type ams_type )
```

This function sets chunk transfer type. This should be called after response of a chunk is received to inform stack that chunked transfer has not been completed.

Parameters

<i>port</i>	Port index
<i>ams_type</i>	Type of PD Atomic Message Sequence that is running.

Returns

ccg_status_t

6.19.2.38 dpm_set_delay_src_cap_start()

```
void dpm_set_delay_src_cap_start (
    uint8_t port,
    uint16_t delay )
```

This function facilitates to delay the starting of source cap on typeC attach. Applicable only when cable is not present.

Parameters

<i>port</i>	Port index.
<i>delay</i>	Delay in ms.

Returns

None.

6.19.2.39 dpm_set_fault_active()

```
ccg_status_t dpm_set_fault_active (
    uint8_t port )
```

This function sets internal flags to indicate that any fault (OVP/OCP/OTP etc.) is active.

Parameters

<i>port</i>	Port index
-------------	------------

Returns

ccg_status_t

6.19.2.40 dpm_set_rp_detach_detect_disabled()

```
void dpm_set_rp_detach_detect_disabled (
    uint8_t port,
    bool is_disabled )
```

Function to update the Rp based source detach detection support for the Type-C port.

Parameters

<i>port</i>	Type-C port index.
<i>is_disabled</i>	Whether Rp based detach detection should be disabled.

6.19.2.41 dpm_sleep()

```
bool dpm_sleep (
    void )
```

This function checks if the device policy manager can go into sleep mode.

Returns

Returns true if possible to go into sleep mode, false otherwise.

6.19.2.42 dpm_start()

```
ccg_status_t dpm_start (
    uint8_t port )
```

This function makes the specified port operational.

Parameters

<i>port</i>	Port index.
-------------	-------------

Returns

CCG_STAT_SUCCESS if operation is successful, CCG_STAT_BAD_PARAM if port index is not correct or dpm_init is not done, CCG_STAT_FAILURE if port is disabled.

6.19.2.43 dpm_stop()

```
ccg_status_t dpm_stop (
    uint8_t port )
```

This function stops the port operation. The port will be put into the lowest power state and will not be operational.

Parameters

<i>port</i>	Port index.
-------------	-------------

Returns

CCG_STAT_SUCCESS if operation is successful, CCG_STAT_BAD_PARAM otherwise.

6.19.2.44 dpm_task()

```
ccg_status_t dpm_task (
    uint8_t port )
```

This function runs the device policy manager task for the specified port.

Parameters

<i>port</i>	Port index.
-------------	-------------

Returns

CCG_STAT_SUCCESS if operation is successful, CCG_STAT_BAD_PARAM otherwise.

6.19.2.45 dpm_typec_command()

```
ccg_status_t dpm_typec_command (
    uint8_t port,
    dpm_typec_cmd_t cmd,
    dpm_typec_cmd_cbk_t cmd_cbk )
```

This function provides an interface for the application module to control the Type C interface.

Parameters

<i>port</i>	Port index.
<i>cmd</i>	Command name.
<i>cmd_cbk</i>	Pointer to the callback function.

Returns

Returns

- CCG_STAT_SUCCESS if the command is registered
- CCG_STAT_BUSY if a previous command is still active
- CCG_STAT_CMD_FAILURE if the port is in a disabled state.

6.19.2.46 dpm_typec_deassert_rp_rd()

```
ccg_status_t dpm_typec_deassert_rp_rd (
    uint8_t port,
    uint8_t channel )
```

This function removes Rp and Rd from the specified CC channel.

Parameters

<i>port</i>	Port index.
<i>channel</i>	CC channel.

Returns

CCG_STAT_SUCCESS if operation is successful, CCG_STAT_BAD_PARAM otherwise.

6.19.2.47 dpm_update_def_cable_cap()

```
void dpm_update_def_cable_cap (
    uint16_t def_cur )
```

This function updates the default cable current characteristics. The parameter is used by the stack to determine the maximum current setting allowed when no e-marked cable is present. The default spec limit is 3A. It should be overridden only for specific captive cable solution with a different current capability. The function should be called only once before the DPM is initialized and the parameter is used for all ports on the device. If the function is not called, the default value of 3A is used.

Parameters

<i>def_cur</i>	Default current setting in 10mA unit (3A is represented as 300).
----------------	--

Warning

The function should be called only when for a non-standard solution and has the current capability.

6.19.2.48 dpm_update_ext_snk_cap()

```
void dpm_update_ext_snk_cap (
    uint8_t port,
    uint8_t * buf_p )
```

Update the extended sink capabilities for the PD port.

Parameters

<i>port</i>	PD port to be updated.
<i>buf↔ _p</i>	Pointer to buffer containing extended sink capabilities data.

Returns

None

6.19.2.49 dpm_update_ext_src_cap()

```
void dpm_update_ext_src_cap (
    uint8_t port,
    uint8_t * buf_p )
```

Update the extended source capabilities for the PD port.

Parameters

<i>port</i>	PD port to be updated.
<i>buf↔ _p</i>	Pointer to buffer containing extended source capabilities data.

Returns

None

6.19.2.50 dpm_update_frs_enable()

```
void dpm_update_frs_enable (
    uint8_t port,
    bool frs_rx_en,
    bool frs_tx_en )
```

Enable/disable the PD 3.0 FRS functionality.

Parameters

<i>port</i>	PD port to be updated.
<i>frs_rx_en</i>	Whether FRS receive is to be enabled.
<i>frs_tx_en</i>	Whether FRS transmit is to be enabled.

Returns

None

6.19.2.51 dpm_update_mux_enable_wait_period()

```
void dpm_update_mux_enable_wait_period (
    uint16_t period )
```

Function to specify the delay to be used between the APP_EVT_TYPEC_ATTACH which is used to enable the Data Mux and the turning ON of the power source. By default, the stack does not use any delay. In cases where the MUX used requires time to properly turn ON, this function can be used to delay the VBus turn ON.

Parameters

<i>period</i>	Delay to be provided (in ms) between MUX enable and VBus turning ON.
---------------	--

Returns

None

6.19.2.52 dpm_update_ndiscover_identity_count()

```
void dpm_update_ndiscover_identity_count (
    uint8_t count )
```

Set the cable discovery message count. This function can be used to change the maximum number of Discover Identity messages sent to the EMCA from the default value of nDiscoverIdentityCount.

Parameters

<i>count</i>	Number of Discover Identity messages that need to be sent. Should be non-zero.
--------------	--

Returns

None

6.19.2.53 dpm_update_port_config()

```
ccg_status_t dpm_update_port_config (
    uint8_t port,
    uint8_t role,
    uint8_t dflt_role,
    uint8_t toggle_en,
    uint8_t try_src_snk_en )
```

Change the PD port configuration at runtime.

This function allows changing the PD port configuration parameters like port role, default port role, DRP toggle enable and Try.Src enable at runtime. These changes are only allowed while the corresponding PD port is disabled.

Parameters

<i>port</i>	USB-PD port to be configured.
<i>role</i>	New port role selection (0 = Sink, 1 = Source, 2 = Dual Role).
<i>dflt_role</i>	New default port role selection (0 = Sink, 1 = Source).
<i>toggle_en</i>	New value for DRP toggle enable flag.
<i>try_src_snk_en</i>	New value for Try.SRC/ TRY.SNK enable flag (0 = Both Try.SRC and TRY.SNK are disabled, 1 = Try.SRC is enabled, 2 = TRY.SNK is enabled).

Returns

Returns

- CCG_STAT_SUCCESS if operation is successful
- CCG_STAT_BAD_PARAM if port index is not correct or other parameters are not correct
- CCG_STAT_FAILURE if port is not disabled.

6.19.2.54 dpm_update_port_status()

```
void dpm_update_port_status (
    uint8_t port,
    uint8_t * status_p,
    uint8_t offset,
    uint8_t byte_cnt )
```

Updates the PD port status returned in response to a Get_Status command.

Parameters

<i>port</i>	PD port to be updated.
<i>status_p</i>	Pointer to buffer containing port status.
<i>offset</i>	Number of bytes of offset to be applied while updating the status.
<i>byte_cnt</i>	Number of bytes of status being updated.

Returns

void

6.19.2.55 dpm_update_rp_audio_accessory()

```
void dpm_update_rp_audio_accessory (
    rp_term_t rp_sel )
```

Set the Rp level that the CCGx as source should use when connected to an audio accessory. Default (USB) Rp is the default setting. The same setting will be used on all Type-C ports managed by the CCGx device.

Parameters

<i>rp_sel</i>	Desired Rp setting to be used.
---------------	--------------------------------

Returns

None

6.19.2.56 dpm_update_snk_cap()

```
ccg_status_t dpm_update_snk_cap (
    uint8_t port,
    uint8_t count,
    pd_do_t * pdo )
```

This function updates the sink PDOs at runtime thereby overriding the sink PDOs in the config table.

Parameters

<i>port</i>	Port index.
<i>count</i>	Count of PDOs.
<i>pdo</i>	Pointer to the PDO array.

Returns

CCG_STAT_SUCCESS if operation is successful, CCG_STAT_BAD_PARAM otherwise.

6.19.2.57 dpm_update_snk_cap_mask()

```
ccg_status_t dpm_update_snk_cap_mask (
    uint8_t port,
    uint8_t mask )
```

This function updates the sink PDO mask at runtime thereby overriding the sink PDO mask specified in the config table.

Parameters

<i>port</i>	Port index.
<i>mask</i>	PDO mask.

Returns

CCG_STAT_SUCCESS if operation is successful, CCG_STAT_BAD_PARAM otherwise.

6.19.2.58 dpm_update_snk_max_min()

```
ccg_status_t dpm_update_snk_max_min (
    uint8_t port,
    uint8_t count,
    uint16_t * max_min )
```

This function updates the sink max/min current/power at runtime thereby overriding the sink max/min current/power specified in the config table.

Parameters

<i>port</i>	Port index.
<i>count</i>	Count of PDOs.
<i>max_min</i>	Pointer to max/min cur/power array.

Returns

CCG_STAT_SUCCESS if operation is successful, CCG_STAT_BAD_PARAM otherwise.

6.19.2.59 dpm_update_snk_wait_cap_period()

```
void dpm_update_snk_wait_cap_period (
    uint16_t period )
```

Function to update the tTypeCSnkWaitCap period used by the CCG PD stack. The default tTypeCSnkWaitCap value used by CCG PD stack is 400 ms which is the mid-value of the spec defined range. However, this timer only starts when the firmware has started running; which could be about 250 ms from device power up in some cases. This function allows the user to update the tTypeCSnkWaitCap period based on the worst case start-up delays for the application.

Parameters

<i>period</i>	The tTypeCSnkWaitCap period to be used, in ms.
---------------	--

Returns

None

6.19.2.60 dpm_update_src_cap()

```
ccg_status_t dpm_update_src_cap (
    uint8_t port,
    uint8_t count,
    pd_do_t * pdo )
```

This function updates the source PDOs at runtime thereby overriding the source PDOs in the config table.

Parameters

<i>port</i>	Port index.
<i>count</i>	Count of PDOs.
<i>pdo</i>	Pointer to the PDO array.

Returns

CCG_STAT_SUCCESS if operation is successful, CCG_STAT_BAD_PARAM otherwise.

6.19.2.61 dpm_update_src_cap_mask()

```
ccg_status_t dpm_update_src_cap_mask (
    uint8_t port,
    uint8_t mask )
```

This function updates the source PDO mask at runtime thereby overriding the source PDO mask in the config table.

Parameters

<i>port</i>	Port index.
<i>mask</i>	PDO mask.

Returns

CCG_STAT_SUCCESS if operation is successful, CCG_STAT_BAD_PARAM otherwise.

6.19.2.62 dpm_update_swap_response()

```
ccg_status_t dpm_update_swap_response (
    uint8_t port,
    uint8_t value )
```

Update the SWAP_RESPONSE setting for the device policy manager.

Parameters

<i>port</i>	Port index.
<i>value</i>	New value for swap response.

Returns

CCG_STAT_SUCCESS if operation is successful, CCG_STAT_BAD_PARAM otherwise.

6.19.2.63 dpm_wakeup()

```
bool dpm_wakeup (
    void )
```

This function configures the device policy manager, for all ports, after the system comes out of deepsleep.

Returns

Returns true if successful, 0 otherwise.

6.19.3 Variable Documentation**6.19.3.1 gl_dpm_port_type**

```
port_type_t gl_dpm_port_type[NO_OF_TYPEC_PORTS]
```

Current port type (DFP/UFP) for each PD port.

6.20 pd_common/pd.h File Reference

```
#include <config.h>
#include <stdint.h>
#include <stdbool.h>
#include <stddef.h>
#include <utils.h>
#include <timer_id.h>
#include <srom_config.h>
```

Data Structures

- union [cc_state_t](#)
- union [pd_hdr_t](#)
- struct [pd_hdr_t::PD_HDR](#)
- union [pd_extd_hdr_t](#)
- struct [pd_extd_hdr_t::EXTD_HDR_T](#)
- union [pd_do_t](#)
- struct [pd_do_t::BIST_DO](#)
- struct [pd_do_t::FIXED_SRC](#)
- struct [pd_do_t::VAR_SRC](#)
- struct [pd_do_t::BAT_SRC](#)
- struct [pd_do_t::SRC_GEN](#)
- struct [pd_do_t::FIXED_SNK](#)
- struct [pd_do_t::VAR_SNK](#)
- struct [pd_do_t::BAT_SNK](#)
- struct [pd_do_t::RDO_FIXED_VAR](#)
- struct [pd_do_t::RDO_FIXED_VAR_GIVEBACK](#)
- struct [pd_do_t::RDO_BAT](#)
- struct [pd_do_t::RDO_BAT_GIVEBACK](#)
- struct [pd_do_t::RDO_GEN](#)
- struct [pd_do_t::RDO_GEN_GVB](#)
- struct [pd_do_t::STD_VDM_HDR](#)
- struct [pd_do_t::USTD_VDM_HDR](#)
- struct [pd_do_t::USTD_QC_4_0_HDR](#)
- struct [pd_do_t::QC_4_0_DATA_VDO](#)
- struct [pd_do_t::STD_VDM_ID_HDR](#)
- struct [pd_do_t::STD_CERT_VDO](#)
- struct [pd_do_t::STD_PROD_VDO](#)
- struct [pd_do_t::STD_CBL_VDO](#)
- struct [pd_do_t::PAS_CBL_VDO](#)
- struct [pd_do_t::ACT_CBL_VDO](#)
- struct [pd_do_t::ACT_CBL_VDO_1](#)
- struct [pd_do_t::ACT_CBL_VDO_2](#)
- struct [pd_do_t::STD_AMA_VDO](#)
- struct [pd_do_t::STD_AMA_VDO_PD3](#)
- struct [pd_do_t::STD_SVID_RESP_VDO](#)
- struct [pd_do_t::STD_DP_VDO](#)
- struct [pd_do_t::DP_STATUS_VDO](#)
- struct [pd_do_t::DP_CONFIG_VDO](#)

- struct [pd_do_t::PPS_SRC](#)
- struct [pd_do_t::PPS_SNK](#)
- struct [pd_do_t::RDO_PPS](#)
- struct [pd_do_t::ADO_ALERT](#)
- struct [pd_do_t::TBT_UFP_VDO](#)
- struct [pd_do_t::TBT_VDO](#)
- struct [pd_do_t::TBT_CBL_VDO](#)
- struct [pd_do_t::UFP_VDO_1](#)
- struct [pd_do_t::DFP_VDO](#)
- struct [pd_do_t::ENTERUSB_VDO](#)
- struct [pd_power_status_t](#)
- union [pd_port_status_t](#)
- struct [pd_port_status_t::PD_PORT_STAT](#)
- struct [pd_port_config_t](#)
- struct [custom_host_cfg_settings_t](#)
- struct [icl_tgl_cfg_settings_t](#)
- struct [alt_cfg_settings_t](#)
- struct [custom_alt_cfg_settings_t](#)
- struct [sensor_data_t](#)
- struct [auto_cfg_settings_t](#)
- struct [tbthost_cfg_settings_t](#)
- struct [ovp_settings_t](#)
- struct [ocp_settings_t](#)
- struct [rcp_settings_t](#)
- struct [uvp_settings_t](#)
- struct [scp_settings_t](#)
- struct [vconn_ocp_settings_t](#)
- struct [otp_settings_t](#)
- struct [pwr_params_t](#)
- struct [chg_cfg_params_t](#)
- struct [bat_chg_params_t](#)
- struct [bb_settings_t](#)
- struct [pd_config_t](#)
- struct [app_resp_t](#)
- struct [vdm_resp_t](#)
- struct [dpm_pd_cmd_buf_t](#)
- struct [contract_t](#)
- struct [pd_contract_info_t](#)
- struct [pd_packet_t](#)
- struct [pd_packet_extd_t](#)
- struct [pd_stack_conf_t](#)
- struct [app_cbk_t](#)
- struct [dpm_status_t](#)

Macros

- #define [PD_EXTERNALLY_POWERED_BIT_POS](#) (7u)
- #define [BC_SINK_1_2_MODE_ENABLE_MASK](#) (0x01u)
- #define [BC_SINK_APPLE_MODE_ENABLE_MASK](#) (0x02u)
- #define [BC_SRC_1_2_MODE_ENABLE_MASK](#) (0x01u)
- #define [BC_SRC_APPLE_MODE_ENABLE_MASK](#) (0x02u)
- #define [BC_SRC_QC_MODE_ENABLE_MASK](#) (0x04u)
- #define [BC_SRC_AFC_MODE_ENABLE_MASK](#) (0x08u)
- #define [BC_SRC_QC_4_0_MODE_ENABLE_MASK](#) (0x10u)

- #define BC_SRC_QC_VER_2_CLASS_A_VAL (0u)
- #define BC_SRC_QC_VER_2_CLASS_B_VAL (1u)
- #define BC_SRC_QC_VER_3_CLASS_A_VAL (2u)
- #define BC_SRC_QC_VER_3_CLASS_B_VAL (3u)
- #define PD_MAX_SRC_CAP_TRY (6u)
- #define GIVE_BACK_MASK (0x8000u)
- #define SNK_MIN_MAX_MASK (0x3FFu)
- #define GET_DR_SWAP_RESP(resp) ((resp) & 0x3u)
- #define GET_PR_SWAP_RESP(resp) (((resp) & 0xCu) >> 2u)
- #define GET_VCONN_SWAP_RESP(resp) (((resp) & 0x30u) >> 4u)
- #define MAX_SRC_CAP_COUNT (50u)
- #define MAX_HARD_RESET_COUNT (3u)
- #define MAX_CBL_DSC_ID_COUNT (20u)
- #define MAX_PR_SWAP_WAIT_COUNT (2u)
- #define MAX_NO_OF_DO (7u)
- #define MAX_NO_OF_PDO (MAX_NO_OF_DO)
- #define MAX_NO_OF_VDO (MAX_NO_OF_DO)
- #define VDM_HEADER_IDX (0u)
- #define BDO_HDR_IDX (0u)
- #define ID_HEADER_IDX (1u)
- #define CERT_STAT_IDX (2u)
- #define PRODUCT_VDO_IDX (3u)
- #define PRODUCT_TYPE_VDO_1_IDX (4u)
- #define PRODUCT_TYPE_VDO_2_IDX (5u)
- #define PRODUCT_TYPE_VDO_3_IDX (6u)
- #define RDO_IDX (0u)
- #define MAX_EXTD_PKT_SIZE (260u)
- #define MAX_EXTD_PKT_WORDS (65u)
- #define MAX_EXTD_MSG_LEGACY_LEN (26u)
- #define MAX_MESSAGE_ID (7u)
- #define MAX_SOP_TYPES (3)
- #define STD_SVID (0xFF00u)
- #define DP_SVID (0xFF01u)
- #define TBT_SVID (0x8087u)
- #define UFP_NON_PH_ALT_MODE_SUPP_MASK (0x4u)
- #define CY_VID (0x04B4u)
- #define STD_VDM_VERSION_IDX (13u)
- #define STD_VDM_VERSION_REV3 (1u)
- #define STD_VDM_VERSION_REV2 (0u)
- #define STD_VDM_VERSION (0u)
- #define VSAFE_0V (800u)
- #define VSAFE_3_6V (3600u)
- #define VSAFE_5V (5000u)
- #define VSAFE_9V (9000u)
- #define VSAFE_12V (12000u)
- #define VSAFE_13V (13000u)
- #define VSAFE_15V (15000u)
- #define VSAFE_19V (19000u)
- #define VSAFE_20V (20000u)
- #define VSAFE_0V_PR_SWAP_SNK_SRC (3000u)
- #define VSAFE_0V_HARD_RESET (3000u)
- #define PD_VOLT_PER_UNIT (50u)
- #define PD_VOLT_PER_UNIT_PPS (100u)
- #define PD_CURRENT_PPS_MULTIPLIER (5u)
- #define ISAFE_0A (0u)

- #define [ISAFE_DEF](#) (50u)
- #define [I_0P9A](#) (90u)
- #define [I_1A](#) (100)
- #define [I_1P5A](#) (150)
- #define [I_2A](#) (200)
- #define [I_3A](#) (300)
- #define [I_4A](#) (400)
- #define [I_5A](#) (500)
- #define [PD_CUR_PER_UNIT](#) (10u)
- #define [SNK_DETACH_VBUS_POLL_COUNT](#) (5u)
- #define [DRP_TOGGLE_PERIOD](#) (75u)
- #define [SRC_DRP_MIN_DC](#) (30)
- #define [CC_CHANNEL_1](#) (0u)
- #define [CC_CHANNEL_2](#) (1u)
- #define [TYPEC_FSM_NONE](#) (0x00000000u)
- #define [TYPEC_FSM_GENERIC](#) (0x00000001u)
- #define [HPD_RX_ACTIVITY_TIMER_PERIOD_MIN](#) (5u)
- #define [HPD_RX_ACTIVITY_TIMER_PERIOD_MAX](#) (105u)
- #define [PD_NO_RESPONSE_TIMER_PERIOD](#) (5000u)
- #define [PD_CBL_POWER_UP_TIMER_PERIOD](#) (55u)
- #define [PD_CBL_DSC_ID_TIMER_PERIOD](#) (49u)
- #define [PD_CBL_DSC_ID_START_TIMER_PERIOD](#) (43u)
- #define [PD_CBL_DELAY_TIMER_PERIOD](#) (2u)
- #define [PD_PHY_BUSY_TIMER_PERIOD](#) (15u)
- #define [PD_HARD_RESET_TX_TIMER_PERIOD](#) (20u)
- #define [PD_VCONN_SWAP_INITIATOR_TIMER_PERIOD](#) (110u)
- #define [PD_VCONN_SWAP_INITIATOR_DELAY_PERIOD](#) (500u)
- #define [PD_VBUS_TURN_ON_TIMER_PERIOD](#) (275u)
- #define [PD_VBUS_TURN_OFF_TIMER_PERIOD](#) (625u)
- #define [PD_PS_SRC_TRANS_TIMER_PERIOD](#) (400u)
- #define [PD_PS_SRC_OFF_TIMER_PERIOD](#) (900u)
- #define [PD_PS_SRC_ON_TIMER_PERIOD](#) (450u)
- #define [PD_PS_SNK_TRANSITION_TIMER_PERIOD](#) (500u)
- #define [PD_SRC_RECOVER_TIMER_PERIOD](#) (800u)
- #define [PD_SENDER_RESPONSE_TIMER_PERIOD](#) (27u)
- #define [PD_RECEIVER_RESPONSE_TIMER_PERIOD](#) (15u)
- #define [PD_SINK_WAIT_CAP_TIMER_PERIOD](#) (400u)
- #define [PD_SRC_CAP_TIMER_PERIOD](#) (180u)
- #define [PD_SWAP_SRC_START_TIMER_PERIOD](#) (55u)
- #define [PD_SOURCE_TRANSITION_TIMER_PERIOD](#) (28u)
- #define [PD_VCONN_OFF_TIMER_PERIOD](#) (25u)
- #define [PD_VCONN_ON_TIMER_PERIOD](#) (100u)
- #define [PD_UFP_VCONN_DISCHARGE_DURATION](#) (10u)
- #define [PD_VCONN_SRC_DISC_TIMER_PERIOD](#) (200u)
- #define [PD_VCONN_REAPPLIED_TIMER_PERIOD](#) (18u)
- #define [PD_DATA_RESET_TIMER_PERIOD](#) (220u)
- #define [PD_DATA_RESET_TIMEOUT_PERIOD](#) (250u)
- #define [PD_DATA_RESET_COMPLETION_DELAY](#) (225u)
- #define [PD_VCONN_TURN_ON_TIMER_PERIOD](#) (10u)
- #define [PD_CBL_READY_TIMER_PERIOD](#) (50u)
- #define [PD_VDM_RESPONSE_TIMER_PERIOD](#) (27u)
- #define [PD_VDM_ENTER_MODE_RESPONSE_TIMER_PERIOD](#) (45u)
- #define [PD_VDM_EXIT_MODE_RESPONSE_TIMER_PERIOD](#) (45u)
- #define [PD_DPM_RESP_REC_RESP_PERIOD](#) (20u)
- #define [PD_BIST_CONT_MODE_TIMER_PERIOD](#) (55u)

- #define PD_SINK_VBUS_TURN_OFF_TIMER_PERIOD (750u)
- #define PD_SINK_VBUS_TURN_ON_TIMER_PERIOD (1300u)
- #define PD_PS_HARD_RESET_TIMER_PERIOD (27u)
- #define PD_COLLISION_SRC_COOL_OFF_TIMER_PERIOD (5u)
- #define PD_SINK_TX_TIMER_PERIOD (18u)
- #define PD_PPS_SRC_TIMER_PERIOD (14000u)
- #define TYPEC_CC_DEBOUNCE_TIMER_PERIOD (140u)
- #define TYPEC_PD_DEBOUNCE_TIMER_PERIOD (11u)
- #define TYPEC_RD_DEBOUNCE_TIMER_PERIOD (12u)
- #define TYPEC_ATTACH_WAIT_ENTRY_DELAY_PERIOD (10u)
- #define TYPEC_SRC_DETACH_DEBOUNCE_PERIOD (2u)
- #define TYPEC_PD3_RPCHANGE_DEBOUNCE_PERIOD (2u)
- #define TYPEC_ERROR_RECOVERY_TIMER_PERIOD (50u)
- #define TYPEC_DRP_TRY_TIMER_PERIOD (110u)
- #define TYPEC_TRY_TIMEOUT_PERIOD (800u)
- #define CCG_FRS_TX_ENABLE_MASK (0x02u)
- #define CCG_FRS_RX_ENABLE_MASK (0x01u)
- #define CCG_PD_EXT_SRCCAP_SIZE (24u)
- #define CCG_PD_EXT_SRCCAP_INP_INDEX (21u)
- #define CCG_PD_EXT_SRCCAP_INP_UNCONSTRAINED (0x02u)
- #define CCG_PD_EXT_SRCCAP_PDP_INDEX (23u)
- #define CCG_PD_EXT_SNKCAP_SIZE (21u)
- #define CCG_PD_EXT_SNKCAP_BUF_SIZE (24u)
- #define CCG_PD_EXT_SNKCAP_VERS_INDEX (10u)
- #define CCG_PD_EXT_STATUS_SIZE (6u)
- #define CCG_PD_EXT_PPS_STATUS_SIZE (4u)
- #define CCG_PD_FIX_SRC_PDO_MASK_REV2 (0xFE3FFFFFu)
- #define CCG_PD_FIX_SRC_PDO_MASK_REV3 (0xFF3FFFFFu)
- #define CCG_PD_FLAG_CONTRACT_NEG_ACTIVE (1u)
- #define CCG_PD_FLAG_EXPLICIT_CONTRACT (2u)
- #define CCG_PD_FLAG_SRC_READY (4u)
- #define CCG_PD_FLAG_POWER_SINK (8u)
- #define CCG_CC_STAT_ZOPEN (0u)
- #define CCG_CC_STAT_DRP_TOGGLE (1u)
- #define CCG_CC_STAT_RD_PRESENT (2u)
- #define CCG_CC_STAT_RP_PRESENT (4u)
- #define CCG_CC_STAT_VCONN_ACTIVE (8u)
- #define CCG_DPM_ERROR_NONE (0u)
- #define CCG_DPM_ERROR_NO_VCONN (1u)
- #define CCG_USB4_EUDO_USB4_EN_MASK (0x26000000u)
- #define CCG_USB_MODE_USB4 (2u)
- #define CCG_USB_MODE_USB3 (1u)
- #define CCG_USB_MODE_USB2 (0u)
- #define TBT_GEN_3 (3u)
- #define UFP_VDO_1_RECFG_ALT_MODE_PARAM_MASK (0x20u)
- #define TBT_CTRLR_ALPINE_RIDGE_SGL (0u)
- #define TBT_CTRLR_ALPINE_RIDGE_DUAL (1u)
- #define TBT_CTRLR_TITAN_RIDGE_SGL (2u)
- #define TBT_CTRLR_TITAN_RIDGE_DUAL (3u)
- #define DP_HPD_TYPE_GPIO (0u)
- #define DP_HPD_TYPE_VIRTUAL (1u)
- #define HOST_SBU_CFG_FULL (0u)
- #define HOST_SBU_CFG_FIXED_POL (1u)
- #define HOST_SBU_CFG_PASS_THROUGH (2u)

Typedefs

- typedef `pwr_params_t` `typeA_pwr_params_t`
- typedef `chg_cfg_params_t` `typeA_chg_cfg_params_t`
- typedef void(* `pd_cbk_t`) (uint8_t port, uint32_t event)
- typedef void(* `dpm_pd_cmd_cbk_t`) (uint8_t port, `resp_status_t` resp, const `pd_packet_t` *pkt_ptr)
- typedef void(* `app_resp_cbk_t`) (uint8_t port, `app_resp_t` *resp)
- typedef void(* `vdm_resp_cbk_t`) (uint8_t port, `vdm_resp_t` *resp)
- typedef void(* `pwr_ready_cbk_t`) (uint8_t port)
- typedef void(* `sink_discharge_off_cbk_t`) (uint8_t port)
- typedef void(* `dpm_typec_cmd_cbk_t`) (uint8_t port, `dpm_typec_cmd_resp_t` resp)

Enumerations

- enum `pd_err_recov_reason_t` {
`ERR_RECOV_REASON_NONE` = 0, `ERR_RECOV_HR_FAIL`, `ERR_RECOV_PROTECT_FAULT`,
`ERR_RECOV_POWER_FAULT`,
`ERR_RECOV_BAD_DATA_ROLE`, `ERR_RECOV_FRS_FAIL`, `ERR_RECOV_DATA_RESET_FAIL` }
- enum `pd_emca_sr_reason_t` { `EMCA_SR_REASON_NONE` = 0, `EMCA_SR_CABLE_DISC`, `EMCA_SR_ALT_MODE_DISC` }
- enum `pd_cable_reset_reason_t` { `EMCA_CABLE_RES_NONE` = 0, `EMCA_CABLE_RES_SR_TIMEOUT` }
- enum `pd_hard_reset_reason_t` {
`PD_HARDRES_REASON_NONE` = 0, `PD_HARDRES_REASON_NO_SRC_CAP`, `PD_HARDRES_REASON_HOSTCONN`,
`PD_HARDRES_REASON_SR_ERROR`,
`PD_HARDRES_REASON_CONTRACT_ERROR`, `PD_HARDRES_REASON_DR_SWAP`, `PD_HARDRES_REASON_VBUS_O`
`PD_HARDRES_REASON_VBUS_OCP`,
`PD_HARDRES_REASON_AMS_ERROR` }
- enum `pd_soft_reset_reason_t` { `PD_SOFTRES_REASON_NONE` = 0, `PD_SOFTRES_REASON_SRCNEG_ERROR`,
`PD_SOFTRES_REASON_SKNNEG_ERROR`, `PD_SOFTRES_REASON_AMS_ERROR` }
- enum `app_fault_mask_t` {
`CFG_TABLE_OVP_EN_MASK` = 0x01u, `CFG_TABLE_OCP_EN_MASK` = 0x02u, `CFG_TABLE_UVP_EN_MASK`
= 0x04u, `CFG_TABLE_SCP_EN_MASK` = 0x08u,
`CFG_TABLE_VCONN_OCP_EN_MASK` = 0x10u, `CFG_TABLE_OTP_EN_MASK` = 0x20u, `CFG_TABLE_RCP_EN_MASK`
= 0x40u }
- enum `app_swap_resp_t` { `APP_RESP_ACCEPT` = 0u, `APP_RESP_REJECT`, `APP_RESP_WAIT`,
`APP_RESP_NOT_SUPPORTED` }
- enum `pd_rev_t` { `PD_REV1` = 0, `PD_REV2`, `PD_REV3`, `PD_REV_RSVD` }
- enum `pd_msg_class_t` {
`PD_CTRL_MSG` = 0, `PD_DATA_MSG`, `PD_EXTD_MSG`, `PD_CABLE_RESET`,
`PD_MSG_RSVD` }
- enum `rdo_type_t` { `FIXED_VAR_RDO` = 0, `BAT_RDO` }
- enum `ctrl_msg_t` {
`CTRL_MSG_RSRVD` = 0, `CTRL_MSG_GOOD_CRC`, `CTRL_MSG_GO_TO_MIN`, `CTRL_MSG_ACCEPT`,
`CTRL_MSG_REJECT`, `CTRL_MSG_PING`, `CTRL_MSG_PS_RDY`, `CTRL_MSG_GET_SOURCE_CAP`,
`CTRL_MSG_GET_SINK_CAP`, `CTRL_MSG_DR_SWAP`, `CTRL_MSG_PR_SWAP`, `CTRL_MSG_VCONN_SWAP`,
`CTRL_MSG_WAIT`, `CTRL_MSG_SOFT_RESET`, `CTRL_MSG_DATA_RESET`, `CTRL_MSG_DATA_RESET_COMPLETE`,
`CTRL_MSG_NOT_SUPPORTED` = 16, `CTRL_MSG_GET_SRC_CAP_EXTD`, `CTRL_MSG_GET_STATUS`,
`CTRL_MSG_FR_SWAP`,
`CTRL_MSG_GET_PPS_STATUS`, `CTRL_MSG_GET_COUNTRY_CODES`, `CTRL_MSG_GET_SNK_CAP_EXTD`
}
- enum `data_msg_t` {
`DATA_MSG_SRC_CAP` = 1, `DATA_MSG_REQUEST`, `DATA_MSG_BIST`, `DATA_MSG_SNK_CAP`,
`DATA_MSG_BAT_STATUS`, `DATA_MSG_ALERT`, `DATA_MSG_GET_COUNTRY_INFO`, `DATA_MSG_ENTER_USB`,
`DATA_MSG_VDM` = 15 }

- enum `extd_msg_t` {
`EXTD_MSG_SRC_CAP_EXTD = 1, EXTD_MSG_STATUS, EXTD_MSG_GET_BAT_CAP, EXTD_MSG_GET_BAT_STATUS,`
`EXTD_MSG_BAT_CAP, EXTD_MSG_GET_MANU_INFO, EXTD_MSG_MANU_INFO, EXTD_MSG_SECURITY_REQ,`
`EXTD_MSG_SECURITY_RESP, EXTD_MSG_FW_UPDATE_REQ, EXTD_MSG_FW_UPDATE_RESP,`
`EXTD_MSG_PPS_STATUS,`
`EXTD_MSG_COUNTRY_INFO, EXTD_MSG_COUNTRY_CODES, EXTD_MSG_SNK_CAP_EXTD }`
- enum `pdo_t` { `PDO_FIXED_SUPPLY = 0, PDO_BATTERY, PDO_VARIABLE_SUPPLY, PDO_AUGMENTED` }
- enum `apdo_t` { `APDO_PPS = 0, APDO_RSVD1, APDO_RSVD2, APDO_RSVD3` }
- enum `peak_cur_cap_t` { `IMAX_EQ_IOC = 0, IMAX_EQ_130_IOC, IMAX_EQ_150_IOC, IMAX_EQ_200_IOC` }
- enum `bist_mode_t` {
`BIST_RX_MODE = 0, BIST_TX_MODE, BIST_RETURN_COUNTERS_MODE, BIST_CARRIER_MODE_0,`
`BIST_CARRIER_MODE_1, BIST_CARRIER_MODE_2, BIST_CARRIER_MODE_3, BIST_EYE_PATTERN_MODE,`
`BIST_TEST_DATA_MODE }`
- enum `sop_t` {
`SOP = 0, SOP_PRIME, SOP_DPRIME, SOP_P_DEBUG,`
`SOP_DP_DEBUG, HARD_RESET, CABLE_RESET, SOP_INVALID }`
- enum `port_role_t` { `PRT_ROLE_SINK = 0, PRT_ROLE_SOURCE, PRT_DUAL` }
- enum `port_type_t` { `PRT_TYPE_UFP = 0, PRT_TYPE_DFP, PRT_TYPE_DRP` }
- enum `fr_swap_supp_t` { `FR_SWAP_NOT_SUPPORTED = 0, FR_SWAP_DEF_USB, FR_SWAP_1_5A,`
`FR_SWAP_3A }`
- enum `app_req_status_t` {
`REQ_SEND_HARD_RESET = 1, REQ_ACCEPT = 3, REQ_REJECT = 4, REQ_WAIT = 12,`
`REQ_NOT_SUPPORTED = 16 }`
- enum `resp_status_t` {
`SEQ_ABORTED = 0, CMD_FAILED, RES_TIMEOUT, CMD_SENT,`
`RES_RCVD }`
- enum `dpm_pd_cmd_t` {
`DPM_CMD_SRC_CAP_CHNG = 0, DPM_CMD_SNK_CAP_CHNG, DPM_CMD_SEND_GO_TO_MIN,`
`DPM_CMD_GET_SNK_CAP,`
`DPM_CMD_GET_SRC_CAP, DPM_CMD_SEND_HARD_RESET, DPM_CMD_SEND_SOFT_RESET,`
`DPM_CMD_SEND_CABLE_RESET,`
`DPM_CMD_SEND_SOFT_RESET_EMCA, DPM_CMD_SEND_DR_SWAP, DPM_CMD_SEND_PR_SWAP,`
`DPM_CMD_SEND_VCONN_SWAP,`
`DPM_CMD_SEND_VDM, DPM_CMD_SEND_EXTENDED, DPM_CMD_GET_SRC_CAP_EXTENDED,`
`DPM_CMD_GET_STATUS,`
`DPM_CMD_SEND_BATT_STATUS, DPM_CMD_SEND_ALERT, DPM_CMD_SEND_NOT_SUPPORTED,`
`DPM_CMD_INITIATE_CBL_DISCOVERY,`
`DPM_CMD_SEND_DATA_RESET, DPM_CMD_SEND_ENTER_USB, DPM_CMD_GET_SNK_CAP_EXTENDED,`
`DPM_CMD_SEND_INVALID = 0xFFu }`
- enum `pd_devtype_t` {
`DEV_SNK = 1, DEV_SRC, DEV_DBG_ACC, DEV_AUD_ACC,`
`DEV_PWRD_ACC, DEV_VPD, DEV_UNSUPPORTED_ACC }`
- enum `vdm_type_t` { `VDM_TYPE_UNSTRUCTURED = 0, VDM_TYPE_STRUCTURED` }
- enum `std_vdm_cmd_t` {
`VDM_CMD_DSC_IDENTITY = 1, VDM_CMD_DSC_SVIDS, VDM_CMD_DSC_MODES, VDM_CMD_ENTER_MODE,`
`VDM_CMD_EXIT_MODE, VDM_CMD_ATTENTION, VDM_CMD_DP_STATUS_UPDT = 16, VDM_CMD_DP_CONFIGURE`
`= 17 }`
- enum `std_vdm_cmd_type_t` { `CMD_TYPE_INITIATOR = 0, CMD_TYPE_RESP_ACK, CMD_TYPE_RESP_NAK,`
`CMD_TYPE_RESP_BUSY }`
- enum `std_vdm_prod_t` {
`PROD_TYPE_UNDEF = 0, PROD_TYPE_HUB = 1, PROD_TYPE_PERI = 2, PROD_TYPE_PSD = 3,`
`PROD_TYPE_PAS_CBL = 3, PROD_TYPE_ACT_CBL = 4, PROD_TYPE_AMA = 5, PROD_TYPE_VPD =`
`6,`
`PROD_TYPE_RSVD = 7 }`
- enum `std_vdm_ver_t` { `STD_VDM_VER1 = 0, STD_VDM_VER2, STD_VDM_VER3, STD_VDM_VER4` }

- enum `cbl_vbus_cur_t` { `CBL_VBUS_CUR_DFLT` = 0, `CBL_VBUS_CUR_3A`, `CBL_VBUS_CUR_5A`, `CBL_VBUS_CUR_0A` }
- enum `cbl_type_t` { `CBL_TYPE_PASSIVE` = 0, `CBL_TYPE_ACTIVE_RETIMER`, `CBL_TYPE_ACTIVE_REDRIVER`, `CBL_TYPE_OPTICAL` }
- enum `cbl_term_t` { `CBL_TERM_BOTH_PAS_VCONN_NOT_REQ` = 0, `CBL_TERM_BOTH_PAS_VCONN_REQ`, `CBL_TERM_ONE_ACT_ONE_PAS_VCONN_REQ`, `CBL_TERM_BOTH_ACT_VCONN_REQ` }
- enum `usb_sig_supp_t` { `USB_2_0` = 0, `USB_GEN_1`, `USB_GEN_2`, `USB_GEN_3` }
- enum `usb_dev_cap_t` { `DEV_CAP_USB_2_0` = (1<<0), `DEV_CAP_USB_BB_ONLY` = (1<<1), `DEV_CAP_USB_3_2` = (1<<2), `DEV_CAP_USB_4_0` = (1<<3) }
- enum `usb_host_cap_t` { `HOST_CAP_USB_2_0` = (1<<0), `HOST_CAP_USB_3_2` = (1<<1), `HOST_CAP_USB_4_0` = (1<<2) }
- enum `usb_role_t` { `USB_ROLE_DEV` = 0, `USB_ROLE_HOST`, `USB_ROLE_DRD` }
- enum `pe_cbl_state_t` { `CBL_FSM_DISABLED` = 0, `CBL_FSM_ENTRY`, `CBL_FSM_SEND_SOFT_RESET`, `CBL_FSM_SEND_DSC_ID` }
- enum `rp_cc_status_t` { `RP_RA` = 0, `RP_RD`, `RP_OPEN` }
- enum `rd_cc_status_t` { `RD_RA` = 0, `RD_USB`, `RD_1_5A`, `RD_3A`, `RD_ERR` }
- enum `rp_term_t` { `RP_TERM_RP_CUR_DEF` = 0, `RP_TERM_RP_CUR_1_5A`, `RP_TERM_RP_CUR_3A` }
- enum `try_src_snk_t` { `TRY_SRC_TRY_SNK_DISABLED` = 0, `TRY_SRC_ENABLED`, `TRY_SNK_ENABLED` }
- enum `dpm_typec_cmd_t` { `DPM_CMD_SET_RP_DFLT` = 0, `DPM_CMD_SET_RP_1_5A`, `DPM_CMD_SET_RP_3A`, `DPM_CMD_PORT_DISABLE`, `DPM_CMD_TYPEC_ERR_RECOVERY`, `DPM_CMD_TYPEC_INVALID` }
- enum `dpm_typec_cmd_resp_t` { `DPM_RESP_FAIL` = 0, `DPM_RESP_SUCCESS` }
- enum `typec_fsm_state_t` { `TYPEC_FSM_DISABLED` = 0, `TYPEC_FSM_ERR_RECOV`, `TYPEC_FSM_ATTACH_WAIT`, `TYPEC_FSM_TRY_SRC`, `TYPEC_FSM_TRY_WAIT_SNK`, `TYPEC_FSM_TRY_SNK`, `TYPEC_FSM_TRY_WAIT_SRC`, `TYPEC_FSM_UNATTACHED_S`, `TYPEC_FSM_UNATTACHED_SNK`, `TYPEC_FSM_UNATTACHED_WAIT_SRC`, `TYPEC_FSM_AUD_ACC`, `TYPEC_FSM_DBG_ACC`, `TYPEC_FSM_ATTACHED_SRC`, `TYPEC_FSM_ATTACHED_SNK`, `TYPEC_FSM_MAX_STATES` }
- enum `pe_fsm_state_t` { `PE_FSM_OFF` = 0, `PE_FSM_HR_SEND`, `PE_FSM_HR_SRC_TRANS_DFLT`, `PE_FSM_HR_SRC_RECOVER`, `PE_FSM_HR_SRC_VBUS_ON`, `PE_FSM_HR_SNK_TRANS_DFLT`, `PE_FSM_HR_SNK_WAIT_VBUS_OFF`, `PE_FSM_HR_SNK_WAIT_VBUS_ON`, `PE_FSM_BIST_TEST_DATA`, `PE_FSM_BIST_CM2`, `PE_FSM_SNK_STARTUP`, `PE_FSM_SNK_WAIT_FOR_CAP`, `PE_FSM_SNK_EVAL_CAP`, `PE_FSM_SNK_SEL_CAP`, `PE_FSM_SRC_STARTUP`, `PE_FSM_SRC_WAIT_NEW_CAP`, `PE_FSM_SRC_SEND_CBL_SR`, `PE_FSM_SRC_SEND_CBL_DSCID`, `PE_FSM_SRC_SEND_CAP`, `PE_FSM_SRC_DISCOVERY`, `PE_FSM_SRC_NEG_CAP`, `PE_FSM_SRC_TRANS_SUPPLY`, `PE_FSM_SRC_SEND_PS_RDY`, `PE_FSM_SNK_TRANS`, `PE_FSM_SR_SEND`, `PE_FSM_SR_RCVD`, `PE_FSM_VRS_VCONN_ON`, `PE_FSM_VRS_VCONN_OFF`, `PE_FSM_SWAP_EVAL`, `PE_FSM_SWAP_SEND`, `PE_FSM_DRS_CHANGE_ROLE`, `PE_FSM_PRS_SRC_SNK_TRANS`, `PE_FSM_PRS_SRC_SNK_VBUS_OFF`, `PE_FSM_PRS_SRC_SNK_WAIT_PS_RDY`, `PE_FSM_PRS_SNK_SRC_WAIT_PS`, `PE_FSM_PRS_SNK_SRC_VBUS_ON`, `PE_FSM_FRS_CHECK_RP`, `PE_FSM_FRS_SRC_SNK_CC_SIGNAL`, `PE_FSM_READY`, `PE_FSM_SEND_MSG`, `PE_FSM_EVAL_DATA_RESET`, `PE_FSM_SEND_DATA_RESET`, `PE_FSM_EVAL_ENTER_USB`, `PE_FSM_MAX_STATES` }
- enum `pd_contract_status_t` { `PD_CONTRACT_NEGOTIATION_SUCCESSFUL` = 0x01, `PD_CONTRACT_CAP_MISMATCH_DETECTED` = 0x03, `PD_CONTRACT_REJECT_CONTRACT_VALID` = 0x00, `PD_CONTRACT_REJECT_CONTRACT_NOT_VALID` = 0x04, `PD_CONTRACT_REJECT_NO_CONTRACT` = 0x08, `PD_CONTRACT_REJECT_EXPLICIT_CONTRACT` = 0x0C, `PD_CONTRACT_REJECT_NO_EXPLICIT_CONTRACT` = 0x10, `PD_CONTRACT_PS_READY_NOT_RECEIVED` = 0x14, `PD_CONTRACT_PS_READY_NOT_SENT` = 0x18 }
- enum `app_evt_t` { `APP_EVT_UNEXPECTED_VOLTAGE_ON_VBUS`, `APP_EVT_TYPE_C_ERROR_RECOVERY`, `APP_EVT_CONNECT`,

- ```

APP_EVT_DISCONNECT,
APP_EVT_EMCA_DETECTED, APP_EVT_EMCA_NOT_DETECTED, APP_EVT_ALT_MODE, APP_EVT_APP_HW,
APP_EVT_BB, APP_EVT_RP_CHANGE, APP_EVT_HARD_RESET_RCVD, APP_EVT_HARD_RESET_COMPLETE,
APP_EVT_PKT_RCVD, APP_EVT_PR_SWAP_COMPLETE, APP_EVT_DR_SWAP_COMPLETE,
APP_EVT_VCONN_SWAP_COMPLETE,
APP_EVT_SENDER_RESPONSE_TIMEOUT, APP_EVT_VENDOR_RESPONSE_TIMEOUT, APP_EVT_HARD_RESET_SE
APP_EVT_SOFT_RESET_SENT,
APP_EVT_CBL_RESET_SENT, APP_EVT_PE_DISABLED, APP_EVT_PD_CONTRACT_NEGOTIATION_COMPLETE,
APP_EVT_VBUS_OVP_FAULT,
APP_EVT_VBUS_OCP_FAULT, APP_EVT_VCONN_OCP_FAULT, APP_EVT_VBUS_PORT_DISABLE,
APP_EVT_TYPEC_STARTED,
APP_EVT_FR_SWAP_COMPLETE, APP_EVT_TEMPERATURE_FAULT, APP_EVT_HANDLE_EXTENDED_MSG,
APP_EVT_VBUS_UVP_FAULT,
APP_EVT_VBUS_SCP_FAULT, APP_EVT_TYPEC_ATTACH_WAIT, APP_EVT_TYPEC_ATTACH_WAIT_TO_UNATTACHE
APP_EVT_TYPEC_ATTACH,
APP_EVT_CC_OVP, APP_EVT_SBU_OVP, APP_EVT_ALERT_RECEIVED, APP_EVT_SRC_CAP_TRIED_WITH_NO_RES
APP_EVT_PD_SINK_DEVICE_CONNECTED, APP_EVT_VBUS_RCP_FAULT, APP_EVT_STANDBY_CURRENT,
APP_EVT_DATA_RESET_RCVD,
APP_EVT_DATA_RESET_SENT, APP_EVT_DATA_RESET_CPLT, APP_EVT_USB_ENTRY_CPLT,
APP_EVT_DATA_RESET_ACCEPTED,
APP_EVT_CONFIG_ERROR, APP_EVT_POWER_CYCLE, APP_EVT_VBUS_IN_UVP_FAULT, APP_EVT_VBUS_IN_OVP_F
APP_EVT_SYSTEM_OT_FAULT, APP_EVT_CRC_ERROR, APP_EVT_HR_PSRC_ENABLE, APP_EVT_TYPEC_RP_DETAC
APP_EVT_PR_SWAP_ACCEPTED, APP_EVT_HR_SENT_RCVD_DEFERRED, APP_TOTAL_EVENTS }

```
- enum `pd_ams_type` { PD\_AMS\_NONE = 0, PD\_AMS\_NON\_INTR, PD\_AMS\_INTR }
  - enum `vdm_ams_t` { VDM\_AMS\_RESP\_READY = 0, VDM\_AMS\_RESP\_NOT\_REQ, VDM\_AMS\_RESP\_FROM\_EC, VDM\_AMS\_RESP\_NOT\_SUPP }
  - enum `usb_data_sig_t` { USB\_2\_0\_SUPP = 0, USB\_GEN\_1\_SUPP, USB\_GEN\_2\_SUPP, USB\_GEN\_3\_SUPP, USB\_BB\_SUPP, USB\_SIG\_UNKNOWN }
  - enum `data_reset_state_t` { DATA\_RESET\_IDLE = 0, DATA\_RESET\_WAIT\_ACCEPT, DATA\_RESET\_ACCEPTED, DATA\_RESET\_WAIT\_PS\_RDY, DATA\_RESET\_WAIT\_VCONN\_OFF, DATA\_RESET\_SENDING\_PS\_RDY, DATA\_RESET\_WAIT\_VCONN\_ON, DATA\_RESET\_WAIT\_COMPLETION, DATA\_RESET\_COMPLETE\_DELAY }
  - enum `pdo_sel_alg_t` { HIGHEST\_POWER = 1, HIGHEST\_CURRENT, HIGHEST\_VOLTAGE }
  - enum `intel_pf_type_t` { PF\_THUNDERBOLT = 0, PF\_ICE\_LAKE, PF\_TIGER\_LAKE, PF\_MAPLE RIDGE }

## Functions

- const `pd_config_t` \* `get_pd_config` (void)
- const `pd_port_config_t` \* `get_pd_port_config` (uint8\_t port)
- bool `pd_is_msg` (const `pd_packet_t` \*pkt, `sop_t` sop\_type, `pd_msg_class_t` msg\_class, uint32\_t msg\_mask, uint16\_t length)
- `ovp_settings_t` \* `pd_get_ptr_ovp_tbl` (uint8\_t port)
- `ocp_settings_t` \* `pd_get_ptr_ocp_tbl` (uint8\_t port)
- `rcp_settings_t` \* `pd_get_ptr_rcp_tbl` (uint8\_t port)
- `uvp_settings_t` \* `pd_get_ptr_uvp_tbl` (uint8\_t port)
- `scp_settings_t` \* `pd_get_ptr_scp_tbl` (uint8\_t port)
- `vconn_ocp_settings_t` \* `pd_get_ptr_vconn_ocp_tbl` (uint8\_t port)
- `otp_settings_t` \* `pd_get_ptr_otp_tbl` (uint8\_t port)
- `pwr_params_t` \* `pd_get_ptr_pwr_tbl` (uint8\_t port)
- `chg_cfg_params_t` \* `pd_get_ptr_chg_cfg_tbl` (uint8\_t port)
- `bat_chg_params_t` \* `pd_get_ptr_bat_chg_tbl` (uint8\_t port)
- `pwr_params_t` \* `pd_get_ptr_type_a_pwr_tbl` (uint8\_t port)
- `typeA_chg_cfg_params_t` \* `pd_get_ptr_type_a_chg_cfg_tbl` (uint8\_t port)
- `bb_settings_t` \* `pd_get_ptr_bb_tbl` (uint8\_t port)

- [auto\\_cfg\\_settings\\_t](#) \* [pd\\_get\\_ptr\\_auto\\_cfg\\_tbl](#) (uint8\_t port)
- [tbthost\\_cfg\\_settings\\_t](#) \* [pd\\_get\\_ptr\\_tbthost\\_cfg\\_tbl](#) (uint8\_t port)
- [custom\\_host\\_cfg\\_settings\\_t](#) \* [pd\\_get\\_ptr\\_host\\_cfg\\_tbl](#) (uint8\_t port)
- [icl\\_tgl\\_cfg\\_settings\\_t](#) \* [pd\\_get\\_ptr\\_icl\\_tgl\\_cfg\\_tbl](#) (uint8\_t port)

### 6.20.1 Detailed Description

Common definitions and structures used in the CCG USB-PD stack.

Note: Changing values in this header file are not likely to have an impact on the final application behavior; as the pre-defined values would have been compiled into the stack libraries.

### 6.20.2 Macro Definition Documentation

#### 6.20.2.1 BC\_SINK\_1\_2\_MODE\_ENABLE\_MASK

```
#define BC_SINK_1_2_MODE_ENABLE_MASK (0x01u)
```

BC 1.2 sink mode enable mask for config table parameter.

#### 6.20.2.2 BC\_SINK\_APPLE\_MODE\_ENABLE\_MASK

```
#define BC_SINK_APPLE_MODE_ENABLE_MASK (0x02u)
```

Apple sink mode enable mask for config table parameter.

#### 6.20.2.3 BC\_SRC\_1\_2\_MODE\_ENABLE\_MASK

```
#define BC_SRC_1_2_MODE_ENABLE_MASK (0x01u)
```

BC 1.2 Source mode enable mask for config table parameter.

#### 6.20.2.4 BC\_SRC\_AFC\_MODE\_ENABLE\_MASK

```
#define BC_SRC_AFC_MODE_ENABLE_MASK (0x08u)
```

AFC source mode enable mask for config table parameter.

#### 6.20.2.5 BC\_SRC\_APPLE\_MODE\_ENABLE\_MASK

```
#define BC_SRC_APPLE_MODE_ENABLE_MASK (0x02u)
```

Apple source mode enable mask for config table parameter.

#### 6.20.2.6 BC\_SRC\_QC\_4\_0\_MODE\_ENABLE\_MASK

```
#define BC_SRC_QC_4_0_MODE_ENABLE_MASK (0x10u)
```

QC 4.0 mode enable mask for config table parameter.

#### 6.20.2.7 BC\_SRC\_QC\_MODE\_ENABLE\_MASK

```
#define BC_SRC_QC_MODE_ENABLE_MASK (0x04u)
```

QC source mode enable mask for config table parameter.

#### 6.20.2.8 BC\_SRC\_QC\_VER\_2\_CLASS\_A\_VAL

```
#define BC_SRC_QC_VER_2_CLASS_A_VAL (0u)
```

QC source Version and class mask for config table parameter.

#### 6.20.2.9 BC\_SRC\_QC\_VER\_2\_CLASS\_B\_VAL

```
#define BC_SRC_QC_VER_2_CLASS_B_VAL (1u)
```

QC source Version and class mask for config table parameter.

#### 6.20.2.10 BC\_SRC\_QC\_VER\_3\_CLASS\_A\_VAL

```
#define BC_SRC_QC_VER_3_CLASS_A_VAL (2u)
```

QC source Version and class mask for config table parameter.

#### 6.20.2.11 BC\_SRC\_QC\_VER\_3\_CLASS\_B\_VAL

```
#define BC_SRC_QC_VER_3_CLASS_B_VAL (3u)
```

QC source Version and class mask for config table parameter.

#### 6.20.2.12 BDO\_HDR\_IDX

```
#define BDO_HDR_IDX (0u)
```

Index of BIST header data object in a received message.

#### 6.20.2.13 CC\_CHANNEL\_1

```
#define CC_CHANNEL_1 (0u)
```

Reference to CC\_1 pin in the Type-C connector.

#### 6.20.2.14 CC\_CHANNEL\_2

```
#define CC_CHANNEL_2 (1u)
```

Reference to CC\_2 pin in the Type-C connector.

#### 6.20.2.15 CCG\_CC\_STAT\_DRP\_TOGGLE

```
#define CCG_CC_STAT_DRP_TOGGLE (1u)
```

CC line status: DRP toggle in progress.

**6.20.2.16 CCG\_CC\_STAT\_RD\_PRESENT**

```
#define CCG_CC_STAT_RD_PRESENT (2u)
```

CC line status: Rd is applied.

**6.20.2.17 CCG\_CC\_STAT\_RP\_PRESENT**

```
#define CCG_CC_STAT_RP_PRESENT (4u)
```

CC line status: Rp is applied.

**6.20.2.18 CCG\_CC\_STAT\_VCONN\_ACTIVE**

```
#define CCG_CC_STAT_VCONN_ACTIVE (8u)
```

CC line status: VConn is applied.

**6.20.2.19 CCG\_CC\_STAT\_ZOPEN**

```
#define CCG_CC_STAT_ZOPEN (0u)
```

CC line status: ZOpen.

**6.20.2.20 CCG\_DPM\_ERROR\_NO\_VCONN**

```
#define CCG_DPM_ERROR_NO_VCONN (1u)
```

DPM command failed due to absence of VConn.

**6.20.2.21 CCG\_DPM\_ERROR\_NONE**

```
#define CCG_DPM_ERROR_NONE (0u)
```

No additional DPM error information available.

**6.20.2.22 CCG\_FRS\_RX\_ENABLE\_MASK**

```
#define CCG_FRS_RX_ENABLE_MASK (0x01u)
```

FRS receive enable flag in config table setting.

**6.20.2.23 CCG\_FRS\_TX\_ENABLE\_MASK**

```
#define CCG_FRS_TX_ENABLE_MASK (0x02u)
```

FRS transmit enable flag in config table setting.

**6.20.2.24 CCG\_PD\_EXT\_PPS\_STATUS\_SIZE**

```
#define CCG_PD_EXT_PPS_STATUS_SIZE (4u)
```

Size of PPS status extended message in bytes.

**6.20.2.25 CCG\_PD\_EXT\_SNKCAP\_BUF\_SIZE**

```
#define CCG_PD_EXT_SNKCAP_BUF_SIZE (24u)
```

Size of buffer allocated for extended sink cap data.

**6.20.2.26 CCG\_PD\_EXT\_SNKCAP\_SIZE**

```
#define CCG_PD_EXT_SNKCAP_SIZE (21u)
```

Size of extended sink cap message in bytes.

**6.20.2.27 CCG\_PD\_EXT\_SNKCAP\_VERS\_INDEX**

```
#define CCG_PD_EXT_SNKCAP_VERS_INDEX (10u)
```

Offset of SKEDB version field in Ext. Snk Cap. This field must be non-zero for a valid response.

**6.20.2.28 CCG\_PD\_EXT\_SRCCAP\_INP\_INDEX**

```
#define CCG_PD_EXT_SRCCAP_INP_INDEX (21u)
```

Index of Source Inputs field in extended source caps.

**6.20.2.29 CCG\_PD\_EXT\_SRCCAP\_INP\_UNCONSTRAINED**

```
#define CCG_PD_EXT_SRCCAP_INP_UNCONSTRAINED (0x02u)
```

Mask for unconstrained source input in extended source caps.

**6.20.2.30 CCG\_PD\_EXT\_SRCCAP\_PDP\_INDEX**

```
#define CCG_PD_EXT_SRCCAP_PDP_INDEX (23u)
```

Index of PDP field in extended source caps.

**6.20.2.31 CCG\_PD\_EXT\_SRCCAP\_SIZE**

```
#define CCG_PD_EXT_SRCCAP_SIZE (24u)
```

Size of extended source capabilities message in bytes.

**6.20.2.32 CCG\_PD\_EXT\_STATUS\_SIZE**

```
#define CCG_PD_EXT_STATUS_SIZE (6u)
```

Size of status extended message in bytes.

**6.20.2.33 CCG\_PD\_FIX\_SRC\_PDO\_MASK\_REV2**

```
#define CCG_PD_FIX_SRC_PDO_MASK_REV2 (0xFE3FFFFFFu)
```

Mask to be applied on Fixed Supply Source PDO for PD Rev 2.0



**6.20.2.34 CCG\_PD\_FIX\_SRC\_PDO\_MASK\_REV3**

```
#define CCG_PD_FIX_SRC_PDO_MASK_REV3 (0xFF3FFFFFFu)
```

Mask to be applied on Fixed Supply Source PDO for PD Rev 3.0

**6.20.2.35 CCG\_PD\_FLAG\_CONTRACT\_NEG\_ACTIVE**

```
#define CCG_PD_FLAG_CONTRACT_NEG_ACTIVE (1u)
```

Status field indicating that contract negotiation is in progress.

**6.20.2.36 CCG\_PD\_FLAG\_EXPLICIT\_CONTRACT**

```
#define CCG_PD_FLAG_EXPLICIT_CONTRACT (2u)
```

Status field indicating that explicit contract is present.

**6.20.2.37 CCG\_PD\_FLAG\_POWER\_SINK**

```
#define CCG_PD_FLAG_POWER_SINK (8u)
```

Status field indicating that the port is currently a sink.

**6.20.2.38 CCG\_PD\_FLAG\_SRC\_READY**

```
#define CCG_PD_FLAG_SRC_READY (4u)
```

Status field indicating that source is ready.

**6.20.2.39 CCG\_USB4\_EUDO\_USB4\_EN\_MASK**

```
#define CCG_USB4_EUDO_USB4_EN_MASK (0x26000000u)
```

Enter USB DO USB4 mode mask.

**6.20.2.40 CCG\_USB\_MODE\_USB2**

```
#define CCG_USB_MODE_USB2 (0u)
```

USB 2.0 data mode as defined in Enter USB DO.

**6.20.2.41 CCG\_USB\_MODE\_USB3**

```
#define CCG_USB_MODE_USB3 (1u)
```

USB 3.2 data mode as defined in Enter USB DO.

**6.20.2.42 CCG\_USB\_MODE\_USB4**

```
#define CCG_USB_MODE_USB4 (2u)
```

USB4 data mode as defined in Enter USB DO.

**6.20.2.43 CERT\_STAT\_IDX**

```
#define CERT_STAT_IDX (2u)
```

Index of CERT\_STAT data object in a received VDM.

**6.20.2.44 CY\_VID**

```
#define CY_VID (0x04B4u)
```

Cypress VID defined by Cypress for field upgrades.

**6.20.2.45 DP\_HPD\_TYPE\_GPIO**

```
#define DP_HPD_TYPE_GPIO (0u)
```

GPIO based HPD configuration.

**6.20.2.46 DP\_HPD\_TYPE\_VIRTUAL**

```
#define DP_HPD_TYPE_VIRTUAL (1u)
```

Virtual (I2C based) HPD configuration.

**6.20.2.47 DP\_SVID**

```
#define DP_SVID (0xFF01u)
```

Displayport SVID defined by VESA specification.

**6.20.2.48 DRP\_TOGGLE\_PERIOD**

```
#define DRP_TOGGLE_PERIOD (75u)
```

Minimum DRP toggling period, in ms. See Table 4-16 of the Type-C spec Rev1.

**6.20.2.49 GET\_DR\_SWAP\_RESP**

```
#define GET_DR_SWAP_RESP(
 resp) ((resp) & 0x3u)
```

Macro to extract the default DR\_SWAP command response from the swap\_response field in the configuration table.

**6.20.2.50 GET\_PR\_SWAP\_RESP**

```
#define GET_PR_SWAP_RESP(
 resp) (((resp) & 0xCu) >> 2u)
```

Macro to extract the default PR\_SWAP command response from the swap\_response field in the configuration table.

**6.20.2.51 GET\_VCONN\_SWAP\_RESP**

```
#define GET_VCONN_SWAP_RESP(
 resp) (((resp) & 0x30u) >> 4u)
```

Macro to extract the default VCONN\_SWAP command response from the swap\_response field in the configuration table.

#### 6.20.2.52 GIVE\_BACK\_MASK

```
#define GIVE_BACK_MASK (0x8000u)
```

Mask for the give-back supported bit in the snk\_pdo\_max\_min\_current\_pwr field of the configuration table.

#### 6.20.2.53 HOST\_SBU\_CFG\_FIXED\_POL

```
#define HOST_SBU_CFG_FIXED_POL (1u)
```

SBU MUX (AUX/LSXX) connection without polarity switch.

#### 6.20.2.54 HOST\_SBU\_CFG\_FULL

```
#define HOST_SBU_CFG_FULL (0u)
```

Full (AUX/LSXX and polarity) selection for SBU MUX.

#### 6.20.2.55 HOST\_SBU\_CFG\_PASS\_THROUGH

```
#define HOST_SBU_CFG_PASS_THROUGH (2u)
```

Pass through SBU MUX (AUX only) connection.

#### 6.20.2.56 HPD\_RX\_ACTIVITY\_TIMER\_PERIOD\_MAX

```
#define HPD_RX_ACTIVITY_TIMER_PERIOD_MAX (105u)
```

Maximum HPD receiver timer period in ms.

#### 6.20.2.57 HPD\_RX\_ACTIVITY\_TIMER\_PERIOD\_MIN

```
#define HPD_RX_ACTIVITY_TIMER_PERIOD_MIN (5u)
```

Minimum HPD receiver timer period in ms.

#### 6.20.2.58 I\_0P9A

```
#define I_0P9A (90u)
```

VBus current usage = 0.9 A.

#### 6.20.2.59 I\_1A

```
#define I_1A (100)
```

VBus current usage = 1.0 A.

#### 6.20.2.60 I\_1P5A

```
#define I_1P5A (150)
```

VBus current usage = 1.5 A.

**6.20.2.61 I\_2A**

```
#define I_2A (200)
```

VBus current usage = 2.0 A.

**6.20.2.62 I\_3A**

```
#define I_3A (300)
```

VBus current usage = 3.0 A.

**6.20.2.63 I\_4A**

```
#define I_4A (400)
```

VBus current usage = 4.0 A.

**6.20.2.64 I\_5A**

```
#define I_5A (500)
```

VBus current usage = 5.0 A.

**6.20.2.65 ID\_HEADER\_IDX**

```
#define ID_HEADER_IDX (1u)
```

Index of ID\_HEADER data object in a received VDM.

**6.20.2.66 ISAFE\_0A**

```
#define ISAFE_0A (0u)
```

VBus current usage = 0 A.

**6.20.2.67 ISAFE\_DEF**

```
#define ISAFE_DEF (50u)
```

VBus current usage = 0.5 A.

**6.20.2.68 MAX\_CBL\_DSC\_ID\_COUNT**

```
#define MAX_CBL_DSC_ID_COUNT (20u)
```

Maximum number of cable discovery DISCOVER\_IDENTITY messages that should be sent out.

**6.20.2.69 MAX\_EXTD\_MSG\_LEGACY\_LEN**

```
#define MAX_EXTD_MSG_LEGACY_LEN (26u)
```

Maximum legacy Extended message size in bytes.

**6.20.2.70 MAX\_EXTD\_PKT\_SIZE**

```
#define MAX_EXTD_PKT_SIZE (260u)
```

Maximum extended message size in bytes.

**6.20.2.71 MAX\_EXTD\_PKT\_WORDS**

```
#define MAX_EXTD_PKT_WORDS (65u)
```

Maximum extended message 32-bit words. Each word is 32 bit.

**6.20.2.72 MAX\_HARD\_RESET\_COUNT**

```
#define MAX_HARD_RESET_COUNT (3u)
```

Maximum hard reset retry count.

**6.20.2.73 MAX\_MESSAGE\_ID**

```
#define MAX_MESSAGE_ID (7u)
```

Maximum message id value in PD Header.

**6.20.2.74 MAX\_NO\_OF\_DO**

```
#define MAX_NO_OF_DO (7u)
```

Maximum number of DOs in a packet. Limited by PD message definition.

**6.20.2.75 MAX\_NO\_OF\_PDO**

```
#define MAX_NO_OF_PDO (MAX_NO_OF_DO)
```

Maximum number of PDOs in a packet. Limited by PD message definition.

**6.20.2.76 MAX\_NO\_OF\_VDO**

```
#define MAX_NO_OF_VDO (MAX_NO_OF_DO)
```

Maximum number of VDOs in a packet. Limited by PD message definition.

**6.20.2.77 MAX\_PR\_SWAP\_WAIT\_COUNT**

```
#define MAX_PR_SWAP_WAIT_COUNT (2u)
```

Number of PR\_SWAP messages for which DUT sends a WAIT response to allow VConn\_SWAP completion.

**6.20.2.78 MAX\_SOP\_TYPES**

```
#define MAX_SOP_TYPES (3)
```

Max SOP types excluding hard reset, cable reset, SOP\_PDEBUG and SOP\_DPDEBUG.

**6.20.2.79 MAX\_SRC\_CAP\_COUNT**

```
#define MAX_SRC_CAP_COUNT (50u)
```

Maximum retries of Source Capability messages.

**6.20.2.80 PD\_BIST\_CONT\_MODE\_TIMER\_PERIOD**

```
#define PD_BIST_CONT_MODE_TIMER_PERIOD (55u)
```

BIST continuous mode period in ms. See Section 6.5.8.4 of USBPD Spec Rev2 v1.2

**6.20.2.81 PD\_CBL\_DELAY\_TIMER\_PERIOD**

```
#define PD_CBL_DELAY_TIMER_PERIOD (2u)
```

Cable delay timer period in ms. See Section 6.5.14 of USBPD Spec Rev2 v1.2

**6.20.2.82 PD\_CBL\_DSC\_ID\_START\_TIMER\_PERIOD**

```
#define PD_CBL_DSC_ID_START_TIMER_PERIOD (43u)
```

Cable discovery start period in ms. See Section 6.5.15 of USBPD Spec Rev2 v1.2

**6.20.2.83 PD\_CBL\_DSC\_ID\_TIMER\_PERIOD**

```
#define PD_CBL_DSC_ID_TIMER_PERIOD (49u)
```

Cable discovery timer period in ms. See Section 6.5.15 of USBPD Spec Rev2 v1.2

**6.20.2.84 PD\_CBL\_POWER\_UP\_TIMER\_PERIOD**

```
#define PD_CBL_POWER_UP_TIMER_PERIOD (55u)
```

tVconnStable delay required by cable marker to power up.

**6.20.2.85 PD\_CBL\_READY\_TIMER\_PERIOD**

```
#define PD_CBL_READY_TIMER_PERIOD (50u)
```

This timer is the delay between PD startup and sending cable Discover ID request to ensure cable is ready to respond.

**6.20.2.86 PD\_COLLISION\_SRC\_COOL\_OFF\_TIMER\_PERIOD**

```
#define PD_COLLISION_SRC_COOL_OFF_TIMER_PERIOD (5u)
```

Time for which PD 3.0 source will keep Rp as SinkTxNG after returning to Ready state.

**6.20.2.87 PD\_CUR\_PER\_UNIT**

```
#define PD_CUR_PER_UNIT (10u)
```

Current unit used in PDOs.

**6.20.2.88 PD\_CURRENT\_PPS\_MULTIPLIER**

```
#define PD_CURRENT_PPS_MULTIPLIER (5u)
```

Multiplier used to convert from current unit used in other PDOs to that used in PPS PDO/RDO.

**6.20.2.89 PD\_DATA\_RESET\_COMPLETION\_DELAY**

```
#define PD_DATA_RESET_COMPLETION_DELAY (225u)
```

Delay applied before DFP sends Data\_Reset\_Complete message.

**6.20.2.90 PD\_DATA\_RESET\_TIMEOUT\_PERIOD**

```
#define PD_DATA_RESET_TIMEOUT_PERIOD (250u)
```

Data\_Reset\_Complete timeout period.

**6.20.2.91 PD\_DATA\_RESET\_TIMER\_PERIOD**

```
#define PD_DATA_RESET_TIMER_PERIOD (220u)
```

Time between Data\_Reset is accepted and Data\_Reset\_Complete message has to be sent.

**6.20.2.92 PD\_DPM\_RESP\_REC\_RESP\_PERIOD**

```
#define PD_DPM_RESP_REC_RESP_PERIOD (20u)
```

VDM receiver response timer period in ms. See Section 6.5.12.1 of USBPD Spec Rev2 v1.2. This timer is slightly relaxed from the spec value.

**6.20.2.93 PD\_EXTERNALLY\_POWERED\_BIT\_POS**

```
#define PD_EXTERNALLY_POWERED_BIT_POS (7u)
```

Externally powered bit position in Source PDO mask.

**6.20.2.94 PD\_HARD\_RESET\_TX\_TIMER\_PERIOD**

```
#define PD_HARD_RESET_TX_TIMER_PERIOD (20u)
```

Hard reset transmit timer period in ms. See Section 6.3.13 of USBPD Spec Rev2 v1.2

**6.20.2.95 PD\_MAX\_SRC\_CAP\_TRY**

```
#define PD_MAX_SRC_CAP_TRY (6u)
```

MAX SRC CAP retry count used to determine if the device connected is PD Capable or not.

**6.20.2.96 PD\_NO\_RESPONSE\_TIMER\_PERIOD**

```
#define PD_NO_RESPONSE_TIMER_PERIOD (5000u)
```

PD No response timer period in ms. See Section 6.5.7 of USBPD Spec Rev2 v1.2

**6.20.2.97 PD\_PHY\_BUSY\_TIMER\_PERIOD**

```
#define PD_PHY_BUSY_TIMER_PERIOD (15u)
```

Period of timer used internally by stack to prevent PHY lockup in TX state.

**6.20.2.98 PD\_PPS\_SRC\_TIMER\_PERIOD**

```
#define PD_PPS_SRC_TIMER_PERIOD (14000u)
```

PPS timer period in ms.

**6.20.2.99 PD\_PS\_HARD\_RESET\_TIMER\_PERIOD**

```
#define PD_PS_HARD_RESET_TIMER_PERIOD (27u)
```

Hard reset timer period in ms. See Section 6.5.11.2 of USBPD Spec Rev2 v1.2

**6.20.2.100 PD\_PS\_SNK\_TRANSITION\_TIMER\_PERIOD**

```
#define PD_PS_SNK_TRANSITION_TIMER_PERIOD (500u)
```

Snk. transition period in ms. See Section 6.5.6.1 of USBPD Spec Rev2 v1.2

**6.20.2.101 PD\_PS\_SRC\_OFF\_TIMER\_PERIOD**

```
#define PD_PS_SRC_OFF_TIMER_PERIOD (900u)
```

Src. off timer period in ms. See Section 6.5.6.2 of USBPD Spec Rev2 v1.2

**6.20.2.102 PD\_PS\_SRC\_ON\_TIMER\_PERIOD**

```
#define PD_PS_SRC_ON_TIMER_PERIOD (450u)
```

Src. on timer period in ms. See Section 6.5.6.3 of USBPD Spec Rev2 v1.2

**6.20.2.103 PD\_PS\_SRC\_TRANS\_TIMER\_PERIOD**

```
#define PD_PS_SRC_TRANS_TIMER_PERIOD (400u)
```

Src.Trans timer period in ms. See Section 6.5.6.1 of USBPD Spec Rev2 v1.2

**6.20.2.104 PD\_RECEIVER\_RESPONSE\_TIMER\_PERIOD**

```
#define PD_RECEIVER_RESPONSE_TIMER_PERIOD (15u)
```

Receiver response timeout period in ms. See Section 6.5.2 of USBPD Spec Rev2 v1.2

**6.20.2.105 PD\_SENDER\_RESPONSE\_TIMER\_PERIOD**

```
#define PD_SENDER_RESPONSE_TIMER_PERIOD (27u)
```

Sender response timeout period in ms. See Section 6.5.2 of USBPD Spec Rev2 v1.2



**6.20.2.106 PD\_SINK\_TX\_TIMER\_PERIOD**

```
#define PD_SINK_TX_TIMER_PERIOD (18u)
```

Delay between AMS initiation attempts by PD 3.0 sink while Rp = SinkTxNG.

**6.20.2.107 PD\_SINK\_VBUS\_TURN\_OFF\_TIMER\_PERIOD**

```
#define PD_SINK_VBUS_TURN_OFF_TIMER_PERIOD (750u)
```

Time in ms allowed for VBus turn off during hard reset.

**6.20.2.108 PD\_SINK\_VBUS\_TURN\_ON\_TIMER\_PERIOD**

```
#define PD_SINK_VBUS_TURN_ON_TIMER_PERIOD (1300u)
```

Time in ms allowed for VBus to turn on during hard reset.

**6.20.2.109 PD\_SINK\_WAIT\_CAP\_TIMER\_PERIOD**

```
#define PD_SINK_WAIT_CAP_TIMER_PERIOD (400u)
```

Snk wait cap period in ms. See Section 6.5.4.2 of USBPD Spec Rev2 v1.2

**6.20.2.110 PD\_SOURCE\_TRANSITION\_TIMER\_PERIOD**

```
#define PD_SOURCE_TRANSITION_TIMER_PERIOD (28u)
```

Source voltage transition timer period in ms. See Table 7-22 of USBPD Spec Rev2 v1.2

**6.20.2.111 PD\_SRC\_CAP\_TIMER\_PERIOD**

```
#define PD_SRC_CAP_TIMER_PERIOD (180u)
```

Src cap timer period in ms. See Section 6.5.4.1 of USBPD Spec Rev2 v1.2

**6.20.2.112 PD\_SRC\_RECOVER\_TIMER\_PERIOD**

```
#define PD_SRC_RECOVER_TIMER_PERIOD (800u)
```

Src.Recover timer period in ms. See Section 7.6.1 of USBPD Spec Rev2 v1.2

**6.20.2.113 PD\_SWAP\_SRC\_START\_TIMER\_PERIOD**

```
#define PD_SWAP_SRC_START_TIMER_PERIOD (55u)
```

Src start (during PR\_SWAP) period in ms. See Section 6.5.9.2 of USBPD Spec Rev2 v1.2

**6.20.2.114 PD\_UFP\_VCONN\_DISCHARGE\_DURATION**

```
#define PD_UFP_VCONN_DISCHARGE_DURATION (10u)
```

Duration for which the UFP discharges VConn during Data\_Reset sequence.

**6.20.2.115 PD\_VBUS\_TURN\_OFF\_TIMER\_PERIOD**

```
#define PD_VBUS_TURN_OFF_TIMER_PERIOD (625u)
```

VBus OFF timer period in ms. See Table 7-22 of USBPD Spec Rev2 v1.2

**6.20.2.116 PD\_VBUS\_TURN\_ON\_TIMER\_PERIOD**

```
#define PD_VBUS_TURN_ON_TIMER_PERIOD (275u)
```

VBus ON timer period in ms. See Table 7-22 of USBPD Spec Rev2 v1.2

**6.20.2.117 PD\_VCONN\_OFF\_TIMER\_PERIOD**

```
#define PD_VCONN_OFF_TIMER_PERIOD (25u)
```

Vconn off timer period in ms. See Section 6.5.13 of USBPD Spec Rev2 v1.2

**6.20.2.118 PD\_VCONN\_ON\_TIMER\_PERIOD**

```
#define PD_VCONN_ON_TIMER_PERIOD (100u)
```

VConn on timer period in ms.

**6.20.2.119 PD\_VCONN\_REAPPLIED\_TIMER\_PERIOD**

```
#define PD_VCONN_REAPPLIED_TIMER_PERIOD (18u)
```

tVconnReapplied period from USB-PD specification.

**6.20.2.120 PD\_VCONN\_SRC\_DISC\_TIMER\_PERIOD**

```
#define PD_VCONN_SRC_DISC_TIMER_PERIOD (200u)
```

VConnSourceDischarge timer period.

**6.20.2.121 PD\_VCONN\_SWAP\_INITIATOR\_DELAY\_PERIOD**

```
#define PD_VCONN_SWAP_INITIATOR_DELAY_PERIOD (500u)
```

Delay between VConn Swap attempts when 5V supply source is not present.

**6.20.2.122 PD\_VCONN\_SWAP\_INITIATOR\_TIMER\_PERIOD**

```
#define PD_VCONN_SWAP_INITIATOR_TIMER_PERIOD (110u)
```

This timer used by stack to do auto retry VCONN swap before PR swap (if DUT is sink). Minimum gap between VCONN swap request shall be a minimum 100ms, to be safe 110ms is used.

**6.20.2.123 PD\_VCONN\_TURN\_ON\_TIMER\_PERIOD**

```
#define PD_VCONN_TURN_ON_TIMER_PERIOD (10u)
```

Period of VConn monitoring checks done internally.

**6.20.2.124 PD\_VDM\_ENTER\_MODE\_RESPONSE\_TIMER\_PERIOD**

```
#define PD_VDM_ENTER_MODE_RESPONSE_TIMER_PERIOD (45u)
```

Enter mode response timeout period in ms. See Section 6.5.12.2 of USBPD Spec Rev2 v1.2

**6.20.2.125 PD\_VDM\_EXIT\_MODE\_RESPONSE\_TIMER\_PERIOD**

```
#define PD_VDM_EXIT_MODE_RESPONSE_TIMER_PERIOD (45u)
```

Exit mode response timeout period in ms. See Section 6.5.12.3 of USBPD Spec Rev2 v1.2

**6.20.2.126 PD\_VDM\_RESPONSE\_TIMER\_PERIOD**

```
#define PD_VDM_RESPONSE_TIMER_PERIOD (27u)
```

VDM response timer period in ms. See Section 6.5.12.1 of USBPD Spec Rev2 v1.2

**6.20.2.127 PD\_VOLT\_PER\_UNIT**

```
#define PD_VOLT_PER_UNIT (50u)
```

Voltage unit used in PDOs.

**6.20.2.128 PD\_VOLT\_PER\_UNIT\_PPS**

```
#define PD_VOLT_PER_UNIT_PPS (100u)
```

Voltage unit (mV) used in PPS PDOs.

**6.20.2.129 PRODUCT\_TYPE\_VDO\_1\_IDX**

```
#define PRODUCT_TYPE_VDO_1_IDX (4u)
```

Index of the first product type VDO in a VDM.

**6.20.2.130 PRODUCT\_TYPE\_VDO\_2\_IDX**

```
#define PRODUCT_TYPE_VDO_2_IDX (5u)
```

Index of the second product type VDO in a VDM.

**6.20.2.131 PRODUCT\_TYPE\_VDO\_3\_IDX**

```
#define PRODUCT_TYPE_VDO_3_IDX (6u)
```

Index of the third product type VDO in a VDM.

**6.20.2.132 PRODUCT\_VDO\_IDX**

```
#define PRODUCT_VDO_IDX (3u)
```

Index of PRODUCT VDO in a received VDM.

**6.20.2.133 RDO\_IDX**

```
#define RDO_IDX (0u)
```

Index of Request data object in a received message.

**6.20.2.134 SNK\_DETACH\_VBUS\_POLL\_COUNT**

```
#define SNK_DETACH_VBUS_POLL_COUNT (5u)
```

Number of VBus checks used to detect detach as sink.

**6.20.2.135 SNK\_MIN\_MAX\_MASK**

```
#define SNK_MIN_MAX_MASK (0x3FFu)
```

Mask to extract the actual min/max current value from the `snk_pdo_max_min_current_pwr` field of the configuration table.

**6.20.2.136 SRC\_DRP\_MIN\_DC**

```
#define SRC_DRP_MIN_DC (30)
```

Minimum percentage of DRP period for a source. See Table 4-16 of the Type-C spec Rev1.

**6.20.2.137 STD\_SVID**

```
#define STD_SVID (0xFF00u)
```

Standard SVID defined by USB-PD specification.

**6.20.2.138 STD\_VDM\_VERSION**

```
#define STD_VDM_VERSION (0u)
```

Default VDM version used. Corresponds to VDM version 1.0.

**6.20.2.139 STD\_VDM\_VERSION\_IDX**

```
#define STD_VDM_VERSION_IDX (13u)
```

Position of VDM version field in structured VDM header.

**6.20.2.140 STD\_VDM\_VERSION\_REV2**

```
#define STD_VDM_VERSION_REV2 (0u)
```

VDM version 1.0. Used under USB-PD Revision 2.0.

**6.20.2.141 STD\_VDM\_VERSION\_REV3**

```
#define STD_VDM_VERSION_REV3 (1u)
```

VDM version 2.0. Used under USB-PD Revision 3.0.

**6.20.2.142 TBT\_CTRLR\_ALPINE\_RIDGE\_DUAL**

```
#define TBT_CTRLR_ALPINE_RIDGE_DUAL (1u)
```

Dual port Alpine Ridge controller.

**6.20.2.143 TBT\_CTRLR\_ALPINE\_RIDGE\_SGL**

```
#define TBT_CTRLR_ALPINE_RIDGE_SGL (0u)
```

Single port Alpine Ridge controller.

**6.20.2.144 TBT\_CTRLR\_TITAN\_RIDGE\_DUAL**

```
#define TBT_CTRLR_TITAN_RIDGE_DUAL (3u)
```

Dual port Titan Ridge controller.

**6.20.2.145 TBT\_CTRLR\_TITAN\_RIDGE\_SGL**

```
#define TBT_CTRLR_TITAN_RIDGE_SGL (2u)
```

Single port Titan Ridge controller.

**6.20.2.146 TBT\_GEN\_3**

```
#define TBT_GEN_3 (3u)
```

TBT Gen 3 cable identifier in the TBT cable Disc Mode VDO response.

**6.20.2.147 TBT\_SVID**

```
#define TBT_SVID (0x8087u)
```

Thunderbolt SVID defined by Intel specification.

**6.20.2.148 TYPEC\_ATTACH\_WAIT\_ENTRY\_DELAY\_PERIOD**

```
#define TYPEC_ATTACH_WAIT_ENTRY_DELAY_PERIOD (10u)
```

Delay between Attached.Wait state entry and start of checks for detached status.

**6.20.2.149 TYPEC\_CC\_DEBOUNCE\_TIMER\_PERIOD**

```
#define TYPEC_CC_DEBOUNCE_TIMER_PERIOD (140u)
```

CC debounce period in ms from Type-C spec.

**6.20.2.150 TYPEC\_DRP\_TRY\_TIMER\_PERIOD**

```
#define TYPEC_DRP_TRY_TIMER_PERIOD (110u)
```

Type-C Try DRP timer period in ms.

**6.20.2.151 TYPEC\_ERROR\_RECOVERY\_TIMER\_PERIOD**

```
#define TYPEC_ERROR_RECOVERY_TIMER_PERIOD (50u)
```

Type-C error recovery timer period in ms.

**6.20.2.152 TYPEC\_FSM\_GENERIC**

```
#define TYPEC_FSM_GENERIC (0x00000001u)
```

Type-C state machine active mode.

**6.20.2.153 TYPEC\_FSM\_NONE**

```
#define TYPEC_FSM_NONE (0x00000000u)
```

Type-C state machine inactive mode.

**6.20.2.154 TYPEC\_PD3\_RPCHANGE\_DEBOUNCE\_PERIOD**

```
#define TYPEC_PD3_RPCHANGE_DEBOUNCE_PERIOD (2u)
```

Period in ms used to detect Rp change in a PD 3.0 contract.

**6.20.2.155 TYPEC\_PD\_DEBOUNCE\_TIMER\_PERIOD**

```
#define TYPEC_PD_DEBOUNCE_TIMER_PERIOD (11u)
```

PD debounce period in ms.

**6.20.2.156 TYPEC\_RD\_DEBOUNCE\_TIMER\_PERIOD**

```
#define TYPEC_RD_DEBOUNCE_TIMER_PERIOD (12u)
```

Rd debounce period (detach detection) in ms.

**6.20.2.157 TYPEC\_SRC\_DETACH\_DEBOUNCE\_PERIOD**

```
#define TYPEC_SRC_DETACH_DEBOUNCE_PERIOD (2u)
```

Debounce period used to detect detach when we are source.

**6.20.2.158 TYPEC\_TRY\_TIMEOUT\_PERIOD**

```
#define TYPEC_TRY_TIMEOUT_PERIOD (800u)
```

Type-C Try Timeout timer period in ms.

**6.20.2.159 UFP\_NON\_PH\_ALT\_MODE\_SUPP\_MASK**

```
#define UFP_NON_PH_ALT_MODE_SUPP_MASK (0x4u)
```

UFP Supports Non-Physical Alternate Mode UFP VDO1 mask.

**6.20.2.160 UFP\_VDO\_1\_RECFG\_ALT\_MODE\_PARAM\_MASK**

```
#define UFP_VDO_1_RECFG_ALT_MODE_PARAM_MASK (0x20u)
```

TBT Gen 3 cable identifier in the TBT cable Disc Mode VDO response.

**6.20.2.161 VDM\_HEADER\_IDX**

```
#define VDM_HEADER_IDX (0u)
```

Index of VDM header data object in a received message.

**6.20.2.162 VSAFE\_0V**

```
#define VSAFE_0V (800u)
```

Vbus voltage = 0.8 V

**6.20.2.163 VSAFE\_0V\_HARD\_RESET**

```
#define VSAFE_0V_HARD_RESET (3000u)
```

Maximum voltage allowed at the end of a Hard Reset when CCGx is SNK. This is set to 3.0 V.

**6.20.2.164 VSAFE\_0V\_PR\_SWAP\_SNK\_SRC**

```
#define VSAFE_0V_PR_SWAP_SNK_SRC (3000u)
```

Maximum voltage allowed when PS\_RDY is received during a SNK to SRC PR\_SWAP. This is set to 3.0 V.

**6.20.2.165 VSAFE\_12V**

```
#define VSAFE_12V (12000u)
```

Vbus voltage = 12.0 V

**6.20.2.166 VSAFE\_13V**

```
#define VSAFE_13V (13000u)
```

Vbus voltage = 13.0 V

**6.20.2.167 VSAFE\_15V**

```
#define VSAFE_15V (15000u)
```

Vbus voltage = 15.0 V

**6.20.2.168 VSAFE\_19V**

```
#define VSAFE_19V (19000u)
```

Vbus voltage = 19.0 V

**6.20.2.169 VSAFE\_20V**

```
#define VSAFE_20V (20000u)
```

Vbus voltage = 20.0 V

**6.20.2.170 VSAFE\_3\_6V**

```
#define VSAFE_3_6V (3600u)
```

Vbus voltage = 3.6 V

**6.20.2.171 VSAFE\_5V**

```
#define VSAFE_5V (5000u)
```

Vbus voltage = 5.0 V

**6.20.2.172 VSAFE\_9V**

```
#define VSAFE_9V (9000u)
```

Vbus voltage = 9.0 V

**6.20.3 Typedef Documentation****6.20.3.1 app\_resp\_cbk\_t**

```
typedef void(* app_resp_cbk_t) (uint8_t port, app_resp_t *resp)
```

Application response callback. This is the type of callback used by the stack to receive application level responses associated with a PD message such as a SWAP request.

**Parameters**

|             |                                                        |
|-------------|--------------------------------------------------------|
| <i>port</i> | PD port index.                                         |
| <i>resp</i> | Pointer to the structure holding response information. |

**6.20.3.2 dpm\_pd\_cmd\_cbk\_t**

```
typedef void(* dpm_pd_cmd_cbk_t) (uint8_t port, resp_status_t resp, const pd_packet_t *pkt_ptr)
```

DPM PD command callback. This is the type of callback function used by the Policy Engine to report results of a command to the application layer.

**Parameters**

|                |                                                        |
|----------------|--------------------------------------------------------|
| <i>port</i>    | PD port index.                                         |
| <i>resp</i>    | Response code.                                         |
| <i>pkt_ptr</i> | Pointer to any PD packet associated with the response. |



### 6.20.3.3 dpm\_typec\_cmd\_cbk\_t

```
typedef void(* dpm_typec_cmd_cbk_t) (uint8_t port, dpm_typec_cmd_resp_t resp)
```

Type C command response callback. Type of callback used by the stack to report results of a dpm\_typec\_command API call to the application layer.

#### Parameters

|             |                |
|-------------|----------------|
| <i>port</i> | PD port index. |
| <i>resp</i> | Response code. |

### 6.20.3.4 pd\_cbk\_t

```
typedef void(* pd_cbk_t) (uint8_t port, uint32_t event)
```

PD callback prototype. This is a stack internal callback function used by the USB-PD Protocol layer to send events to the Policy Engine. The events notified correspond to Policy Engine events such as HARD RESET or SOFT RESET received.

#### Parameters

|             |                          |
|-------------|--------------------------|
| <i>port</i> | PD port index.           |
| <i>Type</i> | of event being notified. |

### 6.20.3.5 pwr\_ready\_cbk\_t

```
typedef void(* pwr_ready_cbk_t) (uint8_t port)
```

Power ready callback. Type of callback used by the stack to receive notification from the power source/sink hardware manager that the requested power transition has been completed.

#### Parameters

|             |                |
|-------------|----------------|
| <i>port</i> | PD port index. |
|-------------|----------------|

### 6.20.3.6 sink\_discharge\_off\_cbk\_t

```
typedef void(* sink_discharge_off_cbk_t) (uint8_t port)
```

Sink discharge off callback. Callback type used by the stack to receive notification that the sink discharge circuit has been turned off.

#### Parameters

|             |                |
|-------------|----------------|
| <i>port</i> | PD port index. |
|-------------|----------------|

### 6.20.3.7 vdm\_resp\_cbk\_t

```
typedef void(* vdm_resp_cbk_t) (uint8_t port, vdm_resp_t *resp)
```

VDM response callback. This is the type of callback used by the stack to receive application level responses to a VDM received from the port partner or cable marker.

#### Parameters

|             |                                                    |
|-------------|----------------------------------------------------|
| <i>port</i> | PD port index.                                     |
| <i>resp</i> | Pointer to structure holding response information. |

## 6.20.4 Enumeration Type Documentation

### 6.20.4.1 apdo\_t

```
enum apdo_t
```

Enum of the Augmented PDO types.

#### Enumerator

|            |                                |
|------------|--------------------------------|
| APDO_PPS   | Programmable Power Supply PDO. |
| APDO_RSVD1 | Reserved for future use.       |
| APDO_RSVD2 | Reserved for future use.       |
| APDO_RSVD3 | Reserved for future use.       |

### 6.20.4.2 app\_evt\_t

```
enum app_evt_t
```

Enum of events that are signalled to the application.

#### Enumerator

|                                    |                                             |
|------------------------------------|---------------------------------------------|
| APP_EVT_UNEXPECTED_VOLTAGE_ON_VBUS | 0x00: Unexpected high voltage seen on VBus. |
| APP_EVT_TYPE_C_ERROR_RECOVERY      | 0x01: Type-C error recovery initiated.      |
| APP_EVT_CONNECT                    | 0x02: Type-C connect detected.              |
| APP_EVT_DISCONNECT                 | 0x03: Type-C disconnect(detach) detected.   |
| APP_EVT_EMCA_DETECTED              | 0x04: Cable (EMCA) discovery successful.    |
| APP_EVT_EMCA_NOT_DETECTED          | 0x05: Cable (EMCA) discovery timed out.     |
| APP_EVT_ALT_MODE                   | 0x06: Alternate mode related event.         |
| APP_EVT_APP_HW                     | 0x07: MUX control related event.            |
| APP_EVT_BB                         | 0x08: Billboard status change.              |
| APP_EVT_RP_CHANGE                  | 0x09: Rp termination change detected.       |
| APP_EVT_HARD_RESET_RCVD            | 0x0A: Hard Reset received.                  |

## Enumerator

|                                           |                                                                                                           |
|-------------------------------------------|-----------------------------------------------------------------------------------------------------------|
| APP_EVT_HARD_RESET_COMPLETE               | 0x0B: Hard Reset processing completed.                                                                    |
| APP_EVT_PKT_RCVD                          | 0x0C: New PD message received.                                                                            |
| APP_EVT_PR_SWAP_COMPLETE                  | 0x0D: PR_SWAP process completed.                                                                          |
| APP_EVT_DR_SWAP_COMPLETE                  | 0x0E: DR_SWAP process completed.                                                                          |
| APP_EVT_VCONN_SWAP_COMPLETE               | 0x0F: VConn_SWAP process completed.                                                                       |
| APP_EVT_SENDER_RESPONSE_TIMEOUT           | 0x10: Sender response timeout occurred.                                                                   |
| APP_EVT_VENDOR_RESPONSE_TIMEOUT           | 0x11: Vendor message response timeout occurred.                                                           |
| APP_EVT_HARD_RESET_SENT                   | 0x12: Hard Reset sent by CCG.                                                                             |
| APP_EVT_SOFT_RESET_SENT                   | 0x13: Soft Reset sent by CCG.                                                                             |
| APP_EVT_CBL_RESET_SENT                    | 0x14: Cable Reset sent by CCG.                                                                            |
| APP_EVT_PE_DISABLED                       | 0x15: PE.Disabled state entered.                                                                          |
| APP_EVT_PD_CONTRACT_NEGOTIATION_COMPLETED | 0x16: Contract negotiation completed.                                                                     |
| APP_EVT_VBUS_OVP_FAULT                    | 0x17: VBus Over Voltage fault detected.                                                                   |
| APP_EVT_VBUS_OCP_FAULT                    | 0x18: VBus Over Current fault detected.                                                                   |
| APP_EVT_VCONN_OCP_FAULT                   | 0x19: VConn Over Current fault detected.                                                                  |
| APP_EVT_VBUS_PORT_DISABLE                 | 0x1A: PD port disable completed.                                                                          |
| APP_EVT_TYPEC_STARTED                     | 0x1B: PD port enable (start) completed.                                                                   |
| APP_EVT_FR_SWAP_COMPLETE                  | 0x1C: FR_SWAP process completed.                                                                          |
| APP_EVT_TEMPERATURE_FAULT                 | 0x1D: Over Temperature fault detected.                                                                    |
| APP_EVT_HANDLE_EXTENDED_MSG               | 0x1E: Extended message received and needs to be handled.                                                  |
| APP_EVT_VBUS_UVP_FAULT                    | 0x1F: VBus Under Voltage fault detected.                                                                  |
| APP_EVT_VBUS_SCP_FAULT                    | 0x20: VBus Short Circuit fault detected.                                                                  |
| APP_EVT_TYPEC_ATTACH_WAIT                 | 0x21: Type-C AttachWait state entered. For internal use only.                                             |
| APP_EVT_TYPEC_ATTACH_WAIT_TO_UNATTACHED   | 0x22: Type-C transition from AttachWait to Unattached. For internal use only.                             |
| APP_EVT_TYPEC_ATTACH                      | 0x23: Type-C attach event.                                                                                |
| APP_EVT_CC_OVP                            | 0x24: Over Voltage on CC/VConn line detected.                                                             |
| APP_EVT_SBU_OVP                           | 0x25: Over Voltage on SBU1/SBU2 line detected.                                                            |
| APP_EVT_ALERT_RECEIVED                    | 0x26: Alert message received. For internal use only.                                                      |
| APP_EVT_SRC_CAP_TRIED_WITH_NO_RESPONSE    | 0x27: Src Cap tried with no response. For internal use only.                                              |
| APP_EVT_PD_SINK_DEVICE_CONNECTED          | 0x28: Sink device connected. For internal use only.                                                       |
| APP_EVT_VBUS_RCP_FAULT                    | 0x29: VBus Reverse Current fault detected.                                                                |
| APP_EVT_STANDBY_CURRENT                   | 0x2A: Standby Current.                                                                                    |
| APP_EVT_DATA_RESET_RCVD                   | 0x2B: USB4 Data Reset message received. USB connection should be disabled by DFP on receiving this event. |
| APP_EVT_DATA_RESET_SENT                   | 0x2C: USB4 Data Reset message sent. USB connection should be disabled by DFP on receiving this event.     |
| APP_EVT_DATA_RESET_CPLT                   | 0x2D: USB4 Data Reset process complete. No handling required.                                             |
| APP_EVT_USB_ENTRY_CPLT                    | 0x2E: USB4 entry process complete.                                                                        |
| APP_EVT_DATA_RESET_ACCEPTED               | 0x2F: USB4 Data Reset Accepted. USB connection can be re-enabled by DFP on receiving this event.          |
| APP_EVT_CONFIG_ERROR                      | 0x30: Configuration table error event.                                                                    |

## Enumerator

|                               |                                                                                                           |
|-------------------------------|-----------------------------------------------------------------------------------------------------------|
| APP_EVT_POWER_CYCLE           | 0x31: Power cycle / Reset event.                                                                          |
| APP_EVT_VBUS_IN_UVP_FAULT     | 0x32: Vbus_in undervoltage fault detected.                                                                |
| APP_EVT_VBUS_IN_OVP_FAULT     | 0x33: Vbus_in overvoltage fault detected.                                                                 |
| APP_EVT_SYSTEM_OT_FAULT       | 0x34: System overtemperature fault detected.                                                              |
| APP_EVT_CRC_ERROR             | 0x35: PD CRC error detected.                                                                              |
| APP_EVT_HR_PSRC_ENABLE        | 0x36: PSRC enable is about to be called after Hard reset.                                                 |
| APP_EVT_TYPEC_RP_DETACH       | 0x37: Rp removal detected while in the Attached.SNK state.                                                |
| APP_EVT_PR_SWAP_ACCEPTED      | 0x38: PR-SWAP accepted by CCG or port partner.                                                            |
| APP_EVT_HR_SENT_RCVD_DEFERRED | 0x39: Deferred Hard Reset Sent/Received event handling to accommodate retimer communication delay timing. |
| APP_TOTAL_EVENTS              | 0x3A: Total number of application events.                                                                 |

## 6.20.4.3 app\_fault\_mask\_t

enum `app_fault_mask_t`

Fault handling enable/disable masks for use in the configuration table. The respective fault handling will only be enabled if the corresponding bit is set in the `protection_enable` bit in the configuration table.

## Enumerator

|                             |                                                  |
|-----------------------------|--------------------------------------------------|
| CFG_TABLE_OVP_EN_MASK       | VBUS Over-Voltage fault handling enable mask.    |
| CFG_TABLE_OCP_EN_MASK       | VBUS Over-Current fault handling enable mask.    |
| CFG_TABLE_UVP_EN_MASK       | VBUS Under-Voltage fault handling enable mask.   |
| CFG_TABLE_SCP_EN_MASK       | VBUS Short-Circuit fault handling enable mask.   |
| CFG_TABLE_VCONN_OCP_EN_MASK | VCONN Over-Current fault handling enable mask.   |
| CFG_TABLE_OTP_EN_MASK       | Over-Temperature fault handling enable mask.     |
| CFG_TABLE_RCP_EN_MASK       | VBUS Reverse-Current fault handling enable mask. |

## 6.20.4.4 app\_req\_status\_t

enum `app_req_status_t`

Enum of the PD Request results. Enum fields map to the control message field in the PD spec.

## Enumerator

|                     |                                                                           |
|---------------------|---------------------------------------------------------------------------|
| REQ_SEND_HARD_RESET | Invalid message. Send Hard Reset.                                         |
| REQ_ACCEPT          | Send Accept message.                                                      |
| REQ_REJECT          | Send Reject message.                                                      |
| REQ_WAIT            | Send Wait message.                                                        |
| REQ_NOT_SUPPORTED   | Send Not_Supported message. Will translate to Reject message under PD 2.0 |

## 6.20.4.5 app\_swap\_resp\_t

```
enum app_swap_resp_t
```

Possible responses to various USB-PD swap requests from the application layer. The PD stack hands the requests up to the application for handling and gets directions on handling the request in the form of these response codes.

## Enumerator

|                        |                                                               |
|------------------------|---------------------------------------------------------------|
| APP_RESP_ACCEPT        | Swap request should be accepted.                              |
| APP_RESP_REJECT        | Swap request should be rejected.                              |
| APP_RESP_WAIT          | Swap request handling should be delayed (send wait response). |
| APP_RESP_NOT_SUPPORTED | Swap request is not supported.                                |

## 6.20.4.6 bist\_mode\_t

```
enum bist_mode_t
```

Enum of the BIST modes.

## Enumerator

|                           |                                       |
|---------------------------|---------------------------------------|
| BIST_RX_MODE              | BIST receiver mode.                   |
| BIST_TX_MODE              | BIST transmit mode.                   |
| BIST_RETURN_COUNTERS_MODE | Send Returned BIST counters response. |
| BIST_CARRIER_MODE_0       | BIST carrier mode 0.                  |
| BIST_CARRIER_MODE_1       | BIST carrier mode 1.                  |
| BIST_CARRIER_MODE_2       | BIST carrier mode 2.                  |
| BIST_CARRIER_MODE_3       | BIST carrier mode 3.                  |
| BIST_EYE_PATTERN_MODE     | BIST eye pattern.                     |
| BIST_TEST_DATA_MODE       | BIST test data mode.                  |

## 6.20.4.7 cbl\_term\_t

```
enum cbl_term_t
```

Enum of the cable termination types.

## Enumerator

|                                    |                                                  |
|------------------------------------|--------------------------------------------------|
| CBL_TERM_BOTH_PAS_VCONN_NOT_REQ    | Passive cable, VConn not required.               |
| CBL_TERM_BOTH_PAS_VCONN_REQ        | Passive cable, VConn required.                   |
| CBL_TERM_ONE_ACT_ONE_PAS_VCONN_REQ | One end active, one end passive, VConn required. |
| CBL_TERM_BOTH_ACT_VCONN_REQ        | Both ends of cable are active, VConn required.   |

6.20.4.8 `cbl_type_t`enum `cbl_type_t`

Enum of the cable types.

## Enumerator

|                                       |                           |
|---------------------------------------|---------------------------|
| <code>CBL_TYPE_PASSIVE</code>         | Passive cable.            |
| <code>CBL_TYPE_ACTIVE_RETIMER</code>  | Active Re-timer cable.    |
| <code>CBL_TYPE_ACTIVE_REDRIVER</code> | Active Re-driver cable.   |
| <code>CBL_TYPE_OPTICAL</code>         | Optically Isolated cable. |

6.20.4.9 `cbl_vbus_cur_t`enum `cbl_vbus_cur_t`

Enum of the cable current levels.

## Enumerator

|                                |                                            |
|--------------------------------|--------------------------------------------|
| <code>CBL_VBUS_CUR_DFLT</code> | Cable can support a maximum of 900mA.      |
| <code>CBL_VBUS_CUR_3A</code>   | Cable can support a maximum of 3A.         |
| <code>CBL_VBUS_CUR_5A</code>   | Cable can support a maximum of 5A.         |
| <code>CBL_VBUS_CUR_0A</code>   | Cable does not conduct VBus power through. |

6.20.4.10 `ctrl_msg_t`enum `ctrl_msg_t`

Enum of the control message types.

## Enumerator

|                                      |                               |
|--------------------------------------|-------------------------------|
| <code>CTRL_MSG_RSRVD</code>          | 0x00: Reserved message code.  |
| <code>CTRL_MSG_GOOD_CRC</code>       | 0x01: GoodCRC message.        |
| <code>CTRL_MSG_GO_TO_MIN</code>      | 0x02: GotoMin message.        |
| <code>CTRL_MSG_ACCEPT</code>         | 0x03: Accept message.         |
| <code>CTRL_MSG_REJECT</code>         | 0x04: Reject message.         |
| <code>CTRL_MSG_PING</code>           | 0x05: Ping message.           |
| <code>CTRL_MSG_PS_RDY</code>         | 0x06: PS_RDY message.         |
| <code>CTRL_MSG_GET_SOURCE_CAP</code> | 0x07: Get_Source_Cap message. |
| <code>CTRL_MSG_GET_SINK_CAP</code>   | 0x08: Get_Sink_Cap message.   |
| <code>CTRL_MSG_DR_SWAP</code>        | 0x09: DR_Swap message.        |
| <code>CTRL_MSG_PR_SWAP</code>        | 0x0A: PR_Swap message.        |
| <code>CTRL_MSG_VCONN_SWAP</code>     | 0x0B: VCONN_Swap message.     |
| <code>CTRL_MSG_WAIT</code>           | 0x0C: Wait message.           |
| <code>CTRL_MSG_SOFT_RESET</code>     | 0x0D: Soft_Reset message.     |
| <code>CTRL_MSG_DATA_RESET</code>     | 0x0E: Data_Reset message.     |

## Enumerator

|                              |                                        |
|------------------------------|----------------------------------------|
| CTRL_MSG_DATA_RESET_COMPLETE | 0x0F: Data_Reset_Complete message.     |
| CTRL_MSG_NOT_SUPPORTED       | 0x10: Not_Supported message.           |
| CTRL_MSG_GET_SRC_CAP_EXTD    | 0x11: Get_Source_Cap_Extended message. |
| CTRL_MSG_GET_STATUS          | 0x12: Get_Status message .             |
| CTRL_MSG_FR_SWAP             | 0x13: FR_Swap message.                 |
| CTRL_MSG_GET_PPS_STATUS      | 0x14: Get_PPS_Status message.          |
| CTRL_MSG_GET_COUNTRY_CODES   | 0x15: Get_Country_Codes message.       |
| CTRL_MSG_GET_SNK_CAP_EXTD    | 0x16: Get_Sink_Cap_Extended message.   |

## 6.20.4.11 data\_msg\_t

enum [data\\_msg\\_t](#)

Enum of the data message types.

## Enumerator

|                           |                                    |
|---------------------------|------------------------------------|
| DATA_MSG_SRC_CAP          | 0x01: Source_Capabilities message. |
| DATA_MSG_REQUEST          | 0x02: Request message.             |
| DATA_MSG_BIST             | 0x03: BIST message.                |
| DATA_MSG_SNK_CAP          | 0x04: Sink_Capabilities message.   |
| DATA_MSG_BAT_STATUS       | 0x05: Battery_Status message.      |
| DATA_MSG_ALERT            | 0x06: Alert message.               |
| DATA_MSG_GET_COUNTRY_INFO | 0x07: Get_Country_Info message.    |
| DATA_MSG_ENTER_USB        | 0x08: Enter_USB message.           |
| DATA_MSG_VDM              | 0x0F: Vendor_Defined message.      |

## 6.20.4.12 data\_reset\_state\_t

enum [data\\_reset\\_state\\_t](#)

Enumeration of sub-states associated with Data Reset AMS.

## Enumerator

|                            |                                                              |
|----------------------------|--------------------------------------------------------------|
| DATA_RESET_IDLE            | No Data_Reset related operation pending.                     |
| DATA_RESET_WAIT_ACCEPT     | Sender waiting for Data_Reset Acceptance.                    |
| DATA_RESET_ACCEPTED        | Waiting for next step after sending Accept response.         |
| DATA_RESET_WAIT_PS_RDY     | Waiting for PS_RDY at the end of Data_Reset handshake.       |
| DATA_RESET_WAIT_VCONN_OFF  | Waiting for VConn turn-off completion before sending PS_RDY. |
| DATA_RESET_SENDING_PS_RDY  | In the process of sending PS_RDY after Data_Reset handshake. |
| DATA_RESET_WAIT_VCONN_ON   | DFP waiting to turn VConn ON.                                |
| DATA_RESET_WAIT_COMPLETION | UFP waiting for Data_Reset completion.                       |
| DATA_RESET_COMPLETE_DELAY  | DFP waiting to send Data_Reset_Complete message.             |

## 6.20.4.13 dpm\_pd\_cmd\_t

enum [dpm\\_pd\\_cmd\\_t](#)

Enum of the DPM (Device Policy Manager) command types.

## Enumerator

|                                |                                                                       |
|--------------------------------|-----------------------------------------------------------------------|
| DPM_CMD_SRC_CAP_CHNG           | 00: Source Caps changed notification. Used to trigger fresh contract. |
| DPM_CMD_SNK_CAP_CHNG           | 01: Sink Caps changed notification. Used to trigger fresh contract.   |
| DPM_CMD_SEND_GO_TO_MIN         | 02: Send GotoMin message to port partner.                             |
| DPM_CMD_GET_SNK_CAP            | 03: Send Get_Sink_Cap message to port partner.                        |
| DPM_CMD_GET_SRC_CAP            | 04: Send Get_Source_Cap message to port partner.                      |
| DPM_CMD_SEND_HARD_RESET        | 05: Send Hard Reset.                                                  |
| DPM_CMD_SEND_SOFT_RESET        | 06: Send Soft Reset to port partner.                                  |
| DPM_CMD_SEND_CABLE_RESET       | 07: Send Cable Reset.                                                 |
| DPM_CMD_SEND_SOFT_RESET_EMCA   | 08: Send Soft Reset to cable marker.                                  |
| DPM_CMD_SEND_DR_SWAP           | 09: Send DR_Swap request.                                             |
| DPM_CMD_SEND_PR_SWAP           | 0A: Send PR_Swap request.                                             |
| DPM_CMD_SEND_VCONN_SWAP        | 0B: Send VCONN_Swap request.                                          |
| DPM_CMD_SEND_VDM               | 0C: Send VDM message.                                                 |
| DPM_CMD_SEND_EXTENDED          | 0D: Send extended data message.                                       |
| DPM_CMD_GET_SRC_CAP_EXTENDED   | 0E: Send Get_Source_Cap_Extended message.                             |
| DPM_CMD_GET_STATUS             | 0F: Send Get_Status message.                                          |
| DPM_CMD_SEND_BATT_STATUS       | 10: Send Battery_Status data message.                                 |
| DPM_CMD_SEND_ALERT             | 11: Send Alert message.                                               |
| DPM_CMD_SEND_NOT_SUPPORTED     | 12: Send Not_Supported message.                                       |
| DPM_CMD_INITIATE_CBL_DISCOVERY | 13: Initiate cable discovery (preceded by VConn Swap if required).    |
| DPM_CMD_SEND_DATA_RESET        | 14: Send a USB4 Data_Reset message.                                   |
| DPM_CMD_SEND_ENTER_USB         | 15: Send a USB4 Enter_USB message to port partner or cable marker.    |
| DPM_CMD_GET_SNK_CAP_EXTENDED   | 16: Send Get_Sink_Cap_Extended message.                               |
| DPM_CMD_SEND_INVALID           | FF: Invalid command code.                                             |

## 6.20.4.14 dpm\_typec\_cmd\_resp\_t

enum [dpm\\_typec\\_cmd\\_resp\\_t](#)

Enum of the DPM (Device Policy Manager) response types.

## Enumerator

|                  |                    |
|------------------|--------------------|
| DPM_RESP_FAIL    | Command failed.    |
| DPM_RESP_SUCCESS | Command succeeded. |



## 6.20.4.15 dpm\_typec\_cmd\_t

enum [dpm\\_typec\\_cmd\\_t](#)

Enum of the DPM (Device Policy Manager) command types that can be initiated through the `dpm_typec_command` API.

See also

[dpm\\_typec\\_command](#)

## Enumerator

|                            |                                            |
|----------------------------|--------------------------------------------|
| DPM_CMD_SET_RP_DFLT        | Command to select Default Rp.              |
| DPM_CMD_SET_RP_1_5A        | Command to select 1.5 A Rp.                |
| DPM_CMD_SET_RP_3A          | Command to select 3 A Rp.                  |
| DPM_CMD_PORT_DISABLE       | Command to disable the USB-PD port.        |
| DPM_CMD_TYPEC_ERR_RECOVERY | Command to initiate Type-C error recovery. |
| DPM_CMD_TYPEC_INVALID      | Invalid command type.                      |

## 6.20.4.16 extd\_msg\_t

enum [extd\\_msg\\_t](#)

Enum of the extended data message types.

## Enumerator

|                         |                                             |
|-------------------------|---------------------------------------------|
| EXTD_MSG_SRC_CAP_EXTD   | 0x01: Source_Capabilities_Extended message. |
| EXTD_MSG_STATUS         | 0x02: Status message.                       |
| EXTD_MSG_GET_BAT_CAP    | 0x03: Get_Battery_Cap message.              |
| EXTD_MSG_GET_BAT_STATUS | 0x04: Get_Battery_Status message.           |
| EXTD_MSG_BAT_CAP        | 0x05: Battery_Capabilities message.         |
| EXTD_MSG_GET_MANU_INFO  | 0x06: Get_Manufacturer_Info message.        |
| EXTD_MSG_MANU_INFO      | 0x07: Manufacturer_Info message.            |
| EXTD_MSG_SECURITY_REQ   | 0x08: Security_Request message.             |
| EXTD_MSG_SECURITY_RESP  | 0x09: Security_Response message.            |
| EXTD_MSG_FW_UPDATE_REQ  | 0x0A: Firmware_Update_Request message.      |
| EXTD_MSG_FW_UPDATE_RESP | 0x0B: Firmware_Update_Response message.     |
| EXTD_MSG_PPS_STATUS     | 0x0C: PPS_Status message.                   |
| EXTD_MSG_COUNTRY_INFO   | 0x0D: Country_Info message.                 |
| EXTD_MSG_COUNTRY_CODES  | 0x0E: Country_Codes message.                |
| EXTD_MSG_SNK_CAP_EXTD   | 0x0F: Sink_Capabilities_Extended message.   |

## 6.20.4.17 fr\_swap\_supp\_t

enum `fr_swap_supp_t`

Enum to hold FR swap options in sink capabilities.

## Enumerator

|                       |                                                             |
|-----------------------|-------------------------------------------------------------|
| FR_SWAP_NOT_SUPPORTED | FR_Swap is not supported.                                   |
| FR_SWAP_DEF_USB       | Device will sink less than 900 mA of current after FR_Swap. |
| FR_SWAP_1_5A          | Device will sink less than 1.5 A of current after FR_Swap.  |
| FR_SWAP_3A            | Device will sink less than 3 A of current after FR_SWAP.    |

## 6.20.4.18 intel\_pf\_type\_t

enum `intel_pf_type_t`

List of Intel TBT/USB platform types in which the CCG device is used.

## Enumerator

|                |                                                         |
|----------------|---------------------------------------------------------|
| PF_THUNDERBOLT | Thunderbolt platforms like Alpine Ridge or Titan Ridge. |
| PF_ICE_LAKE    | Intel IceLake platform.                                 |
| PF_TIGER_LAKE  | Intel TigerLake platform.                               |
| PF_MAPLE_RIDGE | Intel RocketLake + Maple Ridge platform.                |

## 6.20.4.19 pd\_ams\_type

enum `pd_ams_type`

Type of USB-PD Atomic Message Sequence (AMS).

## Enumerator

|                 |                                  |
|-----------------|----------------------------------|
| PD_AMS_NONE     | No AMS active.                   |
| PD_AMS_NON_INTR | Non-interruptible AMS is active. |
| PD_AMS_INTR     | Interruptible AMS is active.     |

## 6.20.4.20 pd\_cable\_reset\_reason\_t

enum `pd_cable_reset_reason_t`

Enumeration of reasons for issuing a Cable Reset.

## Enumerator

|                           |                                    |
|---------------------------|------------------------------------|
| EMCA_CABLE_RES_NONE       | No Cable Reset performed.          |
| EMCA_CABLE_RES_SR_TIMEOUT | SOP' or SOP'' SoftReset timed out. |

## 6.20.4.21 pd\_contract\_status\_t

enum pd\_contract\_status\_t

Enum of possible PD contract negotiation scenarios that are used to signal the application event handler. This status will be reported in byte 0 of the event data passed along with the APP\_EVT\_PD\_CONTRACT\_NEGOTIATION\_COMPLETE event. Bytes 3:1 of the event data are not used; and bytes 7:4 will report the RDO where applicable.

## Enumerator

|                                         |                                                                       |
|-----------------------------------------|-----------------------------------------------------------------------|
| PD_CONTRACT_NEGOTIATION_SUCCESSFUL      | PD contract negotiation successful.                                   |
| PD_CONTRACT_CAP_MISMATCH_DETECTED       | PD contract negotiated, but capability mismatch is present.           |
| PD_CONTRACT_REJECT_CONTRACT_VALID       | Contract rejected by CCG, but previous contract is still valid.       |
| PD_CONTRACT_REJECT_CONTRACT_NOT_VALID   | Contract rejected by CCG and previous contract became invalid.        |
| PD_CONTRACT_REJECT_NO_CONTRACT          | Contract rejected by CCG and there was no previous contract.          |
| PD_CONTRACT_REJECT_EXPLICIT_CONTRACT    | Request rejected by port partner while in previous explicit contract. |
| PD_CONTRACT_REJECT_NO_EXPLICIT_CONTRACT | Request rejected by port partner with no previous explicit contract.  |
| PD_CONTRACT_PS_READY_NOT_RECEIVED       | Failed to receive PS_RDY after Accept.                                |
| PD_CONTRACT_PS_READY_NOT_SENT           | Failed to send PS_RDY after Accept.                                   |

## 6.20.4.22 pd\_devtype\_t

enum pd\_devtype\_t

Enum of the attached device type.

## Enumerator

|                     |                                      |
|---------------------|--------------------------------------|
| DEV_SNK             | Power sink device is attached.       |
| DEV_SRC             | Power source device is attached.     |
| DEV_DBG_ACC         | Debug accessory is attached.         |
| DEV_AUD_ACC         | Audio accessory is attached.         |
| DEV_PWRD_ACC        | Powered accessory is attached.       |
| DEV_VPD             | Vconn powered device is attached.    |
| DEV_UNSUPPORTED_ACC | Unsupported device type is attached. |

## 6.20.4.23 pd\_emca\_sr\_reason\_t

enum pd\_emca\_sr\_reason\_t

Enumeration of possible reasons to issue an EMCA (SOP' or SOP") soft reset.

## Enumerator

|                       |                                            |
|-----------------------|--------------------------------------------|
| EMCA_SR_REASON_NONE   | No EMCA soft-reset in progress.            |
| EMCA_SR_CABLE_DISC    | EMCA soft-reset due to cable discovery.    |
| EMCA_SR_ALT_MODE_DISC | EMCA soft-reset due to alt mode discovery. |

## 6.20.4.24 pd\_err\_recov\_reason\_t

```
enum pd_err_recov_reason_t
```

Enumeration of reasons for error recovery entry.

## Enumerator

|                           |                                                             |
|---------------------------|-------------------------------------------------------------|
| ERR_RECOV_REASON_NONE     | Error recovery is not active.                               |
| ERR_RECOV_HR_FAIL         | Error recovery due to hard reset failure.                   |
| ERR_RECOV_PROTECT_FAULT   | Error recovery due to protection (OVP/OCP) fault.           |
| ERR_RECOV_POWER_FAULT     | Error recovery due to voltage fault.                        |
| ERR_RECOV_BAD_DATA_ROLE   | Error recovery due to bad data role in incoming PD message. |
| ERR_RECOV_FRS_FAIL        | Error recovery due to Fast Role Swap error.                 |
| ERR_RECOV_DATA_RESET_FAIL | Error recovery due to failed Data_Reset sequence.           |

## 6.20.4.25 pd\_hard\_reset\_reason\_t

```
enum pd_hard_reset_reason_t
```

Enumeration of reasons for issuing a hard reset.

## Enumerator

|                                  |                                           |
|----------------------------------|-------------------------------------------|
| PD_HARDRES_REASON_NONE           | HardReset not issued.                     |
| PD_HARDRES_REASON_NO_SRC_CAP     | No Source Capability messages received.   |
| PD_HARDRES_REASON_HOSTCONN       | TBT Host Connect state change.            |
| PD_HARDRES_REASON_SR_ERROR       | SoftReset failed.                         |
| PD_HARDRES_REASON_CONTRACT_ERROR | Power contract failed.                    |
| PD_HARDRES_REASON_DRSWAP         | DR Swap received while in Alternate Mode. |
| PD_HARDRES_REASON_VBUS_OVP       | Over-Voltage condition detected.          |
| PD_HARDRES_REASON_VBUS_OCP       | Over-Current condition detected.          |
| PD_HARDRES_REASON_AMS_ERROR      | PD Atomic Message Sequence error.         |

## 6.20.4.26 pd\_msg\_class\_t

```
enum pd_msg_class_t
```

Enum of the PD message types.

## Enumerator

|                |                         |
|----------------|-------------------------|
| PD_CTRL_MSG    | Control message.        |
| PD_DATA_MSG    | Data message.           |
| PD_EXTD_MSG    | Extended data message.  |
| PD_CABLE_RESET | Cable reset message.    |
| PD_MSG_RSVD    | Undefined message type. |

## 6.20.4.27 pd\_rev\_t

enum `pd_rev_t`

Enumeration of the PD spec revisions.

## Enumerator

|             |                                          |
|-------------|------------------------------------------|
| PD_REV1     | USB-PD spec revision 1.0. Not supported. |
| PD_REV2     | USB-PD spec revision 2.0.                |
| PD_REV3     | USB-PD spec revision 3.0.                |
| PD_REV_RSVD | Undefined USB-PD spec revision.          |

## 6.20.4.28 pd\_soft\_reset\_reason\_t

enum `pd_soft_reset_reason_t`

Enumeration of reasons for issuing a soft reset.

## Enumerator

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| PD_SOFTRES_REASON_NONE         | SoftReset not issued.                           |
| PD_SOFTRES_REASON_SRCNEG_ERROR | Contract negotiation error when CCGx is source. |
| PD_SOFTRES_REASON_SNKNEG_ERROR | Contract negotiation error when CCGx is sink.   |
| PD_SOFTRES_REASON_AMS_ERROR    | PD protocol error.                              |

## 6.20.4.29 pdo\_sel\_alg\_t

enum `pdo_sel_alg_t`

Algorithm selection for pdo evaluation. Only fixed SRC\_PDOs take part for current and voltage algorithms.

## Enumerator

|                 |                                                    |
|-----------------|----------------------------------------------------|
| HIGHEST_POWER   | Algorithm to select contract with highest power.   |
| HIGHEST_CURRENT | Algorithm to select contract with highest current. |
| HIGHEST_VOLTAGE | Algorithm to select contract with highest voltage. |

## 6.20.4.30 pdo\_t

enum pdo\_t

Enum of the PDO types.

## Enumerator

|                     |                                              |
|---------------------|----------------------------------------------|
| PDO_FIXED_SUPPLY    | Fixed (voltage) supply power data object.    |
| PDO_BATTERY         | Battery based power data object.             |
| PDO_VARIABLE_SUPPLY | Variable (voltage) supply power data object. |
| PDO_AUGMENTED       | Augmented power data object.                 |

## 6.20.4.31 pe\_cbl\_state\_t

enum pe\_cbl\_state\_t

Enum of the Policy Engine cable discovery states.

## Enumerator

|                         |                                                         |
|-------------------------|---------------------------------------------------------|
| CBL_FSM_DISABLED        | Cable state machine is inactive.                        |
| CBL_FSM_ENTRY           | Cable state machine starting up.                        |
| CBL_FSM_SEND_SOFT_RESET | Cable state machine sending Soft Reset to cable marker. |
| CBL_FSM_SEND_DSC_ID     | Cable state machine waiting for cable response.         |

## 6.20.4.32 pe\_fsm\_state\_t

enum pe\_fsm\_state\_t

Enumeration of Policy Engine states for a USB-PD port. This is for internal stack usage.

## Warning

The ordering of elements must not be altered unless the state table in the stack source is also updated.

## Enumerator

|                             |                                                              |
|-----------------------------|--------------------------------------------------------------|
| PE_FSM_OFF                  | 00: Policy Engine not started.                               |
| PE_FSM_HR_SEND              | 01: Send HardReset                                           |
| PE_FSM_HR_SRC_TRANS_DFLT    | 02: PE_SRC_Transition_to_default                             |
| PE_FSM_HR_SRC_RECOVER       | 03: Policy Engine waiting for recovery before enabling VBus. |
| PE_FSM_HR_SRC_VBUS_ON       | 04: Policy Engine enabling VBus after Hard Reset completion. |
| PE_FSM_HR_SNK_TRANS_DFLT    | 05: PE_SNK_Transition_to_default                             |
| PE_FSM_HR_SNK_WAIT_VBUS_OFF | 06: Policy Engine waiting for VBus turning off.              |
| PE_FSM_HR_SNK_WAIT_VBUS_ON  | 07: Policy Engine waiting for VBus to turn back on.          |
| PE_FSM_BIST_TEST_DATA       | 08: BIST test data state.                                    |

## Enumerator

|                                |                                                       |
|--------------------------------|-------------------------------------------------------|
| PE_FSM_BIST_CM2                | 09: PE_BIST_Carrier_Mode                              |
| PE_FSM_SNK_STARTUP             | 10: PE_SNK_Startup                                    |
| PE_FSM_SNK_WAIT_FOR_CAP        | 11: PE_SNK_Wait_for_Capabilities                      |
| PE_FSM_SNK_EVAL_CAP            | 12: PE_SNK_Evaluate_Capability                        |
| PE_FSM_SNK_SEL_CAP             | 13: PE_SNK_Select_Capability                          |
| PE_FSM_SRC_STARTUP             | 14: PE_SRC_Startup                                    |
| PE_FSM_SRC_WAIT_NEW_CAP        | 15: PE_SRC_Wait_New_Capabilities                      |
| PE_FSM_SRC_SEND_CBL_SR         | 16: PE_CBL_Soft_Reset                                 |
| PE_FSM_SRC_SEND_CBL_DSCID      | 17: PE_CBL_Get_Identity                               |
| PE_FSM_SRC_SEND_CAP            | 18: PE_SRC_Send_Capabilities                          |
| PE_FSM_SRC_DISCOVERY           | 19: PE_SRC_Discovery                                  |
| PE_FSM_SRC_NEG_CAP             | 20: PE_SRC_Negotiate_Capability                       |
| PE_FSM_SRC_TRANS_SUPPLY        | 21: PE_SRC_Transition_Supply                          |
| PE_FSM_SRC_SEND_PS_RDY         | 22: Policy Engine waiting to send PS_RDY.             |
| PE_FSM_SNK_TRANS               | 23: PE_SNK_Transition_Sink                            |
| PE_FSM_SR_SEND                 | 24: Policy Engine sending Soft Reset.                 |
| PE_FSM_SR_RCVD                 | 25: Policy Engine received Soft Reset.                |
| PE_FSM_VRS_VCONN_ON            | 26: Policy Engine waiting for VConn to turn on.       |
| PE_FSM_VRS_VCONN_OFF           | 27: Policy Engine waiting for VConn to turn off.      |
| PE_FSM_SWAP_EVAL               | 28: Evaluate received swap command.                   |
| PE_FSM_SWAP_SEND               | 29: Waiting to send swap command.                     |
| PE_FSM_DRS_CHANGE_ROLE         | 30: Change data role.                                 |
| PE_FSM_PRS_SRC_SNK_TRANS       | 31: Source to Sink PR_Swap transition start.          |
| PE_FSM_PRS_SRC_SNK_VBUS_OFF    | 32: Initial source waiting for VBus turning off.      |
| PE_FSM_PRS_SRC_SNK_WAIT_PS_RDY | 33: Initial source waiting for PS_RDY.                |
| PE_FSM_PRS_SNK_SRC_WAIT_PS_RDY | 34: Initial sink waiting for PS_RDY.                  |
| PE_FSM_PRS_SNK_SRC_VBUS_ON     | 35: Initial sink turning VBus ON.                     |
| PE_FSM_FRS_CHECK_RP            | 36: Initial sink checking Rp to send FR_Swap message. |
| PE_FSM_FRS_SRC_SNK_CC_SIGNAL   | 37: Initial source sending FR_Swap signal.            |
| PE_FSM_READY                   | 38: PE_Ready state.                                   |
| PE_FSM_SEND_MSG                | 39: Policy Engine sending new AMS.                    |
| PE_FSM_EVAL_DATA_RESET         | 40: Policy Engine Handling Data_Reset request.        |
| PE_FSM_SEND_DATA_RESET         | 41: Policy Engine initiating Data_Reset request.      |
| PE_FSM_EVAL_ENTER_USB          | 42: Policy Engine handling Enter USB request.         |
| PE_FSM_MAX_STATES              | 43: Invalid Policy Engine state.                      |

## 6.20.4.33 peak\_cur\_cap\_t

enum `peak_cur_cap_t`

Enum of Peak Current Capability levels.

## Enumerator

|                 |                                          |
|-----------------|------------------------------------------|
| IMAX_EQ_IOC     | Peak current equal to operating current. |
| IMAX_EQ_130_IOC | Peak current is 1.3x operating current.  |

**Enumerator**

|                 |                                         |
|-----------------|-----------------------------------------|
| IMAX_EQ_150_IOC | Peak current is 1.5x operating current. |
| IMAX_EQ_200_IOC | Peak current is 2x operating current.   |

**6.20.4.34 port\_role\_t**enum `port_role_t`

Enum of the PD port roles.

**Enumerator**

|                 |                                                |
|-----------------|------------------------------------------------|
| PRT_ROLE_SINK   | Power sink                                     |
| PRT_ROLE_SOURCE | Power source                                   |
| PRT_DUAL        | Dual Role Power device: can be source or sink. |

**6.20.4.35 port\_type\_t**enum `port_type_t`

Enum of the PD port types.

**Enumerator**

|              |                                                                |
|--------------|----------------------------------------------------------------|
| PRT_TYPE_UFP | Upstream facing port. USB device or Alternate mode accessory.  |
| PRT_TYPE_DFP | Downstream facing port. USB host or Alternate mode controller. |
| PRT_TYPE_DRP | Dual Role data device: can be UFP or DFP.                      |

**6.20.4.36 rd\_cc\_status\_t**enum `rd_cc_status_t`

Enum of the Rd status when Rd is asserted.

**Enumerator**

|         |                                         |
|---------|-----------------------------------------|
| RD_RA   | CCG has applied Rd. No external Rp.     |
| RD_USB  | CCG has applied Rd. Default Rp present. |
| RD_1_5A | CCG has applied Rd. 1.5A Rp present.    |
| RD_3A   | CCG has applied Rd. 3A Rp present.      |
| RD_ERR  | CCG has applied Rd. Error state.        |



## 6.20.4.37 rdo\_type\_t

enum `rdo_type_t`

Enum of the RDO types.

## Enumerator

|               |                                               |
|---------------|-----------------------------------------------|
| FIXED_VAR_RDO | Fixed or variable supply request data object. |
| BAT_RDO       | Battery request data object.                  |

## 6.20.4.38 resp\_status\_t

enum `resp_status_t`

Enum of the response status to DPM commands.

## Enumerator

|             |                                                             |
|-------------|-------------------------------------------------------------|
| SEQ_ABORTED | PD AMS aborted.                                             |
| CMD_FAILED  | PD AMS failed.                                              |
| RES_TIMEOUT | No response received.                                       |
| CMD_SENT    | PD command has been sent. Response wait may be in progress. |
| RES_RCVD    | Response received.                                          |

## 6.20.4.39 rp\_cc\_status\_t

enum `rp_cc_status_t`

Enum of the Rp status when Rp is asserted.

## Enumerator

|         |                                           |
|---------|-------------------------------------------|
| RP_RA   | CCG has applied Rp. External Ra present.  |
| RP_RD   | CCG has applied Rp. External Rd present.  |
| RP_OPEN | CCG has applied Rp. No external pulldown. |

## 6.20.4.40 rp\_term\_t

enum `rp_term_t`

Enum of the CC termination current levels.

## Enumerator

|                     |                 |
|---------------------|-----------------|
| RP_TERM_RP_CUR_DEF  | Use default Rp. |
| RP_TERM_RP_CUR_1_5A | Use 1.5 A Rp.   |
| RP_TERM_RP_CUR_3A   | Use 3A Rp.      |

6.20.4.41 `sop_t`enum `sop_t`

Enum of the SOP (Start Of Frame) types.

## Enumerator

|              |                                                |
|--------------|------------------------------------------------|
| SOP          | SOP: Used for communication with port partner. |
| SOP_PRIME    | SOP': Cable marker communication.              |
| SOP_DPRIME   | SOP'': Cable marker communication.             |
| SOP_P_DEBUG  | SOP'_Debug                                     |
| SOP_DP_DEBUG | SOP''_Debug                                    |
| HARD_RESET   | Hard Reset                                     |
| CABLE_RESET  | Cable Reset                                    |
| SOP_INVALID  | Undefined ordered set.                         |

6.20.4.42 `std_vdm_cmd_t`enum `std_vdm_cmd_t`

Enum of the standard VDM commands.

## Enumerator

|                        |                                    |
|------------------------|------------------------------------|
| VDM_CMD_DSC_IDENTITY   | Discover Identity command.         |
| VDM_CMD_DSC_SVIDS      | Discover SVIDs command.            |
| VDM_CMD_DSC_MODES      | Discover Modes command.            |
| VDM_CMD_ENTER_MODE     | Enter Mode command.                |
| VDM_CMD_EXIT_MODE      | Exit Mode command.                 |
| VDM_CMD_ATTENTION      | Attention message.                 |
| VDM_CMD_DP_STATUS_UPDT | DisplayPort Status Update message. |
| VDM_CMD_DP_CONFIGURE   | DisplayPort Configure command.     |

6.20.4.43 `std_vdm_cmd_type_t`enum `std_vdm_cmd_type_t`

Enum of the standard VDM command types.

## Enumerator

|                    |                                |
|--------------------|--------------------------------|
| CMD_TYPE_INITIATOR | VDM sent by command initiator. |
| CMD_TYPE_RESP_ACK  | ACK response.                  |
| CMD_TYPE_RESP_NAK  | NAK response.                  |
| CMD_TYPE_RESP_BUSY | BUSY response.                 |

## 6.20.4.44 std\_vdm\_prod\_t

enum `std_vdm_prod_t`

Enum of the standard VDM product types.

## Enumerator

|                   |                              |
|-------------------|------------------------------|
| PROD_TYPE_UNDEF   | Undefined device type.       |
| PROD_TYPE_HUB     | Hub device type.             |
| PROD_TYPE_PERI    | Peripheral device type.      |
| PROD_TYPE_PSD     | Power Sink Device.           |
| PROD_TYPE_PAS_CBL | Passive Cable.               |
| PROD_TYPE_ACT_CBL | Active Cable.                |
| PROD_TYPE_AMA     | Alternate Mode Accessory.    |
| PROD_TYPE_VPD     | Vconn powered device.        |
| PROD_TYPE_RSVD    | Reserved. Shall not be used. |

## 6.20.4.45 std\_vdm\_ver\_t

enum `std_vdm_ver_t`

Enum for the standard VDM version.

## Enumerator

|              |                    |
|--------------|--------------------|
| STD_VDM_VER1 | VDM version<br>1.0 |
| STD_VDM_VER2 | VDM version<br>2.0 |
| STD_VDM_VER3 | VDM version<br>3.0 |
| STD_VDM_VER4 | VDM version<br>4.0 |

## 6.20.4.46 try\_src\_snk\_t

enum `try_src_snk_t`

Enum of the Try Source/ Try Sink options.

## Enumerator

|                          |                               |
|--------------------------|-------------------------------|
| TRY_SRC_TRY_SNK_DISABLED | Try.SRC and Try.SNK disabled. |
| TRY_SRC_ENABLED          | Try.SRC enabled.              |
| TRY_SNK_ENABLED          | Try.SNK enabled.              |

#### 6.20.4.47 `typec_fsm_state_t`

enum `typec_fsm_state_t`

Enum of the Type-C FSM states. This is for internal stack usage.

#### Warning

The ordering of elements must not be altered unless the state table is also updated to match.

#### Enumerator

|                                             |                                         |
|---------------------------------------------|-----------------------------------------|
| <code>TYPE_C_FSM_DISABLED</code>            | Type-C state machine is disabled.       |
| <code>TYPE_C_FSM_ERR_RECOV</code>           | Error Recovery state.                   |
| <code>TYPE_C_FSM_ATTACH_WAIT</code>         | AttachWait.SRC or AttachWait.SNK state. |
| <code>TYPE_C_FSM_TRY_SRC</code>             | Try.SRC state.                          |
| <code>TYPE_C_FSM_TRY_WAIT_SNK</code>        | TryWait.SNK state.                      |
| <code>TYPE_C_FSM_TRY_SNK</code>             | Try.SNK state.                          |
| <code>TYPE_C_FSM_TRY_WAIT_SRC</code>        | TryWait.SRC state.                      |
| <code>TYPE_C_FSM_UNATTACHED_SRC</code>      | Unattached.SRC state.                   |
| <code>TYPE_C_FSM_UNATTACHED_SNK</code>      | Unattached.SNK state.                   |
| <code>TYPE_C_FSM_UNATTACHED_WAIT_SRC</code> | UnattachedWait.SRC state.               |
| <code>TYPE_C_FSM_AUD_ACC</code>             | AudioAccessory state.                   |
| <code>TYPE_C_FSM_DBG_ACC</code>             | DebugAccessory state.                   |
| <code>TYPE_C_FSM_ATTACHED_SRC</code>        | Attached.SRC state.                     |
| <code>TYPE_C_FSM_ATTACHED_SNK</code>        | Attached.SNK state.                     |
| <code>TYPE_C_FSM_MAX_STATES</code>          | Invalid Type-C state.                   |

#### 6.20.4.48 `usb_data_sig_t`

enum `usb_data_sig_t`

Enumeration of USB signalling supported by a device or cable.

#### Enumerator

|                              |                                      |
|------------------------------|--------------------------------------|
| <code>USB_2_0_SUPP</code>    | Only USB 2.0 support.                |
| <code>USB_GEN_1_SUPP</code>  | USB 3.1 Gen1 (5 Gbps) support.       |
| <code>USB_GEN_2_SUPP</code>  | USB 3.1 Gen2 (10 Gbps) support.      |
| <code>USB_GEN_3_SUPP</code>  | USB 4 Gen3 (20 Gbps) support.        |
| <code>USB_BB_SUPP</code>     | USB Billboard device support.        |
| <code>USB_SIG_UNKNOWN</code> | USB data signalling support unknown. |

## 6.20.4.49 usb\_dev\_cap\_t

enum `usb_dev_cap_t`

Enum of the USB device capabilities masks.

## Enumerator

|                     |                                  |
|---------------------|----------------------------------|
| DEV_CAP_USB_2_0     | [USB 2.0] Device Capable.        |
| DEV_CAP_USB_BB_ONLY | Device Capable (Billboard only). |
| DEV_CAP_USB_3_2     | [USB 3.2] Device Capable.        |
| DEV_CAP_USB_4_0     | [USB4] Device Capable.           |

## 6.20.4.50 usb\_host\_cap\_t

enum `usb_host_cap_t`

Enum of the USB host capabilities masks.

## Enumerator

|                       |                         |
|-----------------------|-------------------------|
| HOST_CAP_USB_2↔<br>_0 | [USB 2.0] Host Capable. |
| HOST_CAP_USB_3↔<br>_2 | [USB 3.2] Host Capable. |
| HOST_CAP_USB_4↔<br>_0 | [USB4] Host Capable.    |

## 6.20.4.51 usb\_role\_t

enum `usb_role_t`

Enum of the USB role types.

## Enumerator

|               |                                |
|---------------|--------------------------------|
| USB_ROLE_DEV  | USB data role Device.          |
| USB_ROLE_HOST | USB data role Host.            |
| USB_ROLE_DRD  | USB data role Device and Host. |

## 6.20.4.52 usb\_sig\_supp\_t

enum `usb_sig_supp_t`

Enum of the USB signaling support.

## Enumerator

|         |                 |
|---------|-----------------|
| USB_2_0 | [USB 2.0] only. |
|---------|-----------------|

**Enumerator**

|                |                                    |
|----------------|------------------------------------|
| USB_GEN↔<br>_1 | [USB 3.2] Gen1.                    |
| USB_GEN↔<br>_2 | [USB 3.2] Gen2 and [USB 4.0] Gen2. |
| USB_GEN↔<br>_3 | [USB 4.0] Gen 3.                   |

**6.20.4.53 vdm\_ams\_t**

enum `vdm_ams_t`

Enumeration of application responses to policy manager.

**Enumerator**

|                       |                                |
|-----------------------|--------------------------------|
| VDM_AMS_RESP_READY    | Response is ready              |
| VDM_AMS_RESP_NOT_REQ  | No response required           |
| VDM_AMS_RESP_FROM_EC  | Response will come from EC     |
| VDM_AMS_RESP_NOT_SUPP | Send a NOT_SUPPORTED response. |

**6.20.4.54 vdm\_type\_t**

enum `vdm_type_t`

Enum of the VDM types.

**Enumerator**

|                       |                   |
|-----------------------|-------------------|
| VDM_TYPE_UNSTRUCTURED | Unstructured VDM. |
| VDM_TYPE_STRUCTURED   | Structured VDM.   |

**6.20.5 Function Documentation****6.20.5.1 get\_pd\_config()**

```
const pd_config_t* get_pd_config (
 void)
```

This function gets the config table data.

**Returns**

Returns a pointer to the config table info structure.

**Warning**

The information provided by this API must not be altered by the application.

**6.20.5.2 get\_pd\_port\_config()**

```
const pd_port_config_t* get_pd_port_config (
 uint8_t port)
```

This function gets the configuration information for the specified port.

**Parameters**

|             |             |
|-------------|-------------|
| <i>port</i> | Port index. |
|-------------|-------------|

**Returns**

Returns a pointer to the port specific config table info structure.

**Warning**

The information provided by this API must not be altered by the application.

**6.20.5.3 pd\_get\_ptr\_auto\_cfg\_tbl()**

```
auto_cfg_settings_t* pd_get_ptr_auto_cfg_tbl (
 uint8_t port)
```

Retrieve pointer to Automotive Charger settings from the configuration table.

This function retrieves the Automotive Charger settings that is stored in the configuration table and stores it in the run-time data structures.

**Parameters**

|             |                                                    |
|-------------|----------------------------------------------------|
| <i>port</i> | USB-PD port for which the data is to be retrieved. |
|-------------|----------------------------------------------------|

**Returns**

Pointer to Automotive Charger settings.

**6.20.5.4 pd\_get\_ptr\_bat\_chg\_tbl()**

```
bat_chg_params_t* pd_get_ptr_bat_chg_tbl (
 uint8_t port)
```

Retrieve pointer to battery power parameters from the configuration table.

This function retrieves the battery charging parameters that is stored in the configuration table and stores it in the run-time data structures.

**Parameters**

|             |                                                    |
|-------------|----------------------------------------------------|
| <i>port</i> | USB-PD port for which the data is to be retrieved. |
|-------------|----------------------------------------------------|

**Returns**

Pointer to battery charging parameters.

**6.20.5.5 pd\_get\_ptr\_bb\_tbl()**

```
bb_settings_t* pd_get_ptr_bb_tbl (
 uint8_t port)
```

Retrieve pointer to Billboard settings from the configuration table.

This function retrieves the Billboard settings that is stored in the configuration table and stores it in the run-time data structures.

**Parameters**

|             |                                                    |
|-------------|----------------------------------------------------|
| <i>port</i> | USB-PD port for which the data is to be retrieved. |
|-------------|----------------------------------------------------|

**Returns**

Pointer to Billboard settings.

**6.20.5.6 pd\_get\_ptr\_chg\_cfg\_tbl()**

```
chg_cfg_params_t* pd_get_ptr_chg_cfg_tbl (
 uint8_t port)
```

Retrieve pointer to battery charging parameters from the configuration table.

This function retrieves the legacy charging parameters that is stored in the configuration table and stores it in the run-time data structures.

**Parameters**

|             |                                                    |
|-------------|----------------------------------------------------|
| <i>port</i> | USB-PD port for which the data is to be retrieved. |
|-------------|----------------------------------------------------|

**Returns**

Pointer to legacy charging parameters.

**6.20.5.7 pd\_get\_ptr\_host\_cfg\_tbl()**

```
custom_host_cfg_settings_t* pd_get_ptr_host_cfg_tbl (
 uint8_t port)
```

Retrieve pointer to Custom Host config settings from the configuration table.



This function retrieves the Custom Host settings that is stored in the configuration table and stores it in the run-time data structures.

**Parameters**

|             |                                                    |
|-------------|----------------------------------------------------|
| <i>port</i> | USB-PD port for which the data is to be retrieved. |
|-------------|----------------------------------------------------|

**Returns**

Pointer to Custom Host settings.

**6.20.5.8 pd\_get\_ptr\_icl\_tgl\_cfg\_tbl()**

```
icl_tgl_cfg_settings_t* pd_get_ptr_icl_tgl_cfg_tbl (
 uint8_t port)
```

Retrieve pointer to ICL/TGL config settings from the configuration table.

This function retrieves the ICL/TGL settings that is stored in the configuration table and stores it in the run-time data structures.

**Parameters**

|             |                                                    |
|-------------|----------------------------------------------------|
| <i>port</i> | USB-PD port for which the data is to be retrieved. |
|-------------|----------------------------------------------------|

**Returns**

Pointer to Custom Host settings.

**6.20.5.9 pd\_get\_ptr\_ocp\_tbl()**

```
ocp_settings_t* pd_get_ptr_ocp_tbl (
 uint8_t port)
```

Retrieve pointer to VBus OCP settings from the configuration table.

This function retrieves the VBus OCP settings that is stored in the configuration table and stores it in the run-time data structures.

**Parameters**

|             |                                                    |
|-------------|----------------------------------------------------|
| <i>port</i> | USB-PD port for which the data is to be retrieved. |
|-------------|----------------------------------------------------|

**Returns**

Pointer to VBus OCP settings.

**6.20.5.10 pd\_get\_ptr\_otp\_tbl()**

```
otp_settings_t* pd_get_ptr_otp_tbl (
```

```
uint8_t port)
```

Retrieve pointer to OTP settings from the configuration table.

This function retrieves the OTP settings that is stored in the configuration table and stores it in the run-time data structures.

#### Parameters

|             |                                                    |
|-------------|----------------------------------------------------|
| <i>port</i> | USB-PD port for which the data is to be retrieved. |
|-------------|----------------------------------------------------|

#### Returns

Pointer to OTP settings.

#### 6.20.5.11 pd\_get\_ptr\_ovp\_tbl()

```
ovp_settings_t* pd_get_ptr_ovp_tbl (
 uint8_t port)
```

Retrieve pointer to VBus OVP settings from the configuration table.

This function retrieves the VBus OVP settings that is stored in the configuration table and stores it in the run-time data structures.

#### Parameters

|             |                                                    |
|-------------|----------------------------------------------------|
| <i>port</i> | USB-PD port for which the data is to be retrieved. |
|-------------|----------------------------------------------------|

#### Returns

Pointer to VBus OVP settings.

#### 6.20.5.12 pd\_get\_ptr\_pwr\_tbl()

```
pwr_params_t* pd_get_ptr_pwr_tbl (
 uint8_t port)
```

Retrieve pointer to power parameters from the configuration table.

This function retrieves the power parameters that is stored in the configuration table and stores it in the run-time data structures.

#### Parameters

|             |                                                    |
|-------------|----------------------------------------------------|
| <i>port</i> | USB-PD port for which the data is to be retrieved. |
|-------------|----------------------------------------------------|

#### Returns

Pointer to power parameters.

### 6.20.5.13 pd\_get\_ptr\_rcp\_tbl()

```
rcp_settings_t* pd_get_ptr_rcp_tbl (
 uint8_t port)
```

Retrieve pointer to VBus RCP settings from the configuration table.

This function retrieves the VBus RCP settings that is stored in the configuration table and stores it in the run-time data structures.

#### Parameters

|             |                                                    |
|-------------|----------------------------------------------------|
| <i>port</i> | USB-PD port for which the data is to be retrieved. |
|-------------|----------------------------------------------------|

#### Returns

Pointer to VBus RCP settings.

### 6.20.5.14 pd\_get\_ptr\_scp\_tbl()

```
scp_settings_t* pd_get_ptr_scp_tbl (
 uint8_t port)
```

Retrieve pointer to VBus SCP settings from the configuration table.

This function retrieves the VBus SCP settings that is stored in the configuration table and stores it in the run-time data structures.

#### Parameters

|             |                                                    |
|-------------|----------------------------------------------------|
| <i>port</i> | USB-PD port for which the data is to be retrieved. |
|-------------|----------------------------------------------------|

#### Returns

Pointer to VBus SCP settings.

### 6.20.5.15 pd\_get\_ptr\_tbthost\_cfg\_tbl()

```
tbthost_cfg_settings_t* pd_get_ptr_tbthost_cfg_tbl (
 uint8_t port)
```

Retrieve pointer to Thunderbolt Host configuration settings from the configuration table.

#### Parameters

|             |                                                    |
|-------------|----------------------------------------------------|
| <i>port</i> | USB-PD port for which the data is to be retrieved. |
|-------------|----------------------------------------------------|

#### Returns

Pointer to Thunderbolt host config settings.

#### 6.20.5.16 pd\_get\_ptr\_type\_a\_chg\_cfg\_tbl()

```
typeA_chg_cfg_params_t* pd_get_ptr_type_a_chg_cfg_tbl (
 uint8_t port)
```

Retrieve pointer to TYPE-A battery charging parameters from the configuration table.

This function retrieves the legacy charging parameters that is stored in the configuration table and stores it in the run-time data structures.

##### Parameters

|             |                                                    |
|-------------|----------------------------------------------------|
| <i>port</i> | USB-PD port for which the data is to be retrieved. |
|-------------|----------------------------------------------------|

##### Returns

Pointer to legacy charging parameters.

#### 6.20.5.17 pd\_get\_ptr\_type\_a\_pwr\_tbl()

```
pwr_params_t* pd_get_ptr_type_a_pwr_tbl (
 uint8_t port)
```

Retrieve pointer to power parameters of Type-A port from the configuration table.

This function retrieves the power parameters of Type-A port that is stored in the configuration table and stores it in the run-time data structures.

##### Parameters

|             |                                                    |
|-------------|----------------------------------------------------|
| <i>port</i> | USB-PD port for which the data is to be retrieved. |
|-------------|----------------------------------------------------|

##### Returns

Pointer to power parameters of Type-A port.

#### 6.20.5.18 pd\_get\_ptr\_uvp\_tbl()

```
uvp_settings_t* pd_get_ptr_uvp_tbl (
 uint8_t port)
```

Retrieve pointer to VBus UVP settings from the configuration table.

This function retrieves the VBus UVP settings that is stored in the configuration table and stores it in the run-time data structures.

##### Parameters

|             |                                                    |
|-------------|----------------------------------------------------|
| <i>port</i> | USB-PD port for which the data is to be retrieved. |
|-------------|----------------------------------------------------|

**Returns**

Pointer to VBus UVP settings.

**6.20.5.19 pd\_get\_ptr\_vconn\_ocp\_tbl()**

```
vconn_ocp_settings_t* pd_get_ptr_vconn_ocp_tbl (
 uint8_t port)
```

Retrieve pointer to Vconn OCP settings from the configuration table.

This function retrieves the Vconn OCP settings that is stored in the configuration table and stores it in the run-time data structures.

**Parameters**

|             |                                                    |
|-------------|----------------------------------------------------|
| <i>port</i> | USB-PD port for which the data is to be retrieved. |
|-------------|----------------------------------------------------|

**Returns**

Pointer to Vconn OCP settings.

**6.20.5.20 pd\_is\_msg()**

```
bool pd_is_msg (
 const pd_packet_t * pkt,
 sop_t sop_type,
 pd_msg_class_t msg_class,
 uint32_t msg_mask,
 uint16_t length)
```

This is a utility function to check if the packet is an expected message of a certain class.

**Parameters**

|                  |                                                                                                                                                     |
|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>pkt</i>       | Packet pointer.                                                                                                                                     |
| <i>sop_type</i>  | Sop type to match. Passing SOP_INVALID will skip this check.                                                                                        |
| <i>msg_class</i> | Message class: Control, Data or Extended.                                                                                                           |
| <i>msg_mask</i>  | Message mask is a 32 bit mask. Each bit corresponds to a message type corresponding to the message class. If mask is 0, then this check is skipped. |
| <i>length</i>    | Length corresponds to the 'Number of Data Objects' field in the PD header. If length is 0xFFFF, this check is skipped.                              |

**Returns**

Returns true if packet matches all the conditions, else false.

**6.21 pd\_common/pd\_policy\_engine.h File Reference**

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

## Functions

- void [pe\\_init](#) (uint8\_t port)
- void [pe\\_fsm](#) (uint8\_t port)
- void [pe\\_start](#) (uint8\_t port)
- void [pe\\_stop](#) (uint8\_t port)
- bool [pe\\_is\\_busy](#) (uint8\_t port)
- void [pe\\_clear\\_hard\\_reset\\_count](#) (uint8\_t port)
- bool [get\\_spec\\_rev\\_determined](#) (uint8\_t port)
- void [pe\\_disabled](#) (uint8\_t port)
- void [pe\\_push\\_to\\_buf](#) (uint8\_t port, uint8\_t state)
- uint8\_t \* [get\\_pe\\_state\\_buf](#) (uint8\_t port)
- void [pe\\_get\\_pps\\_status](#) (uint8\_t port, uint8\_t \*pps\_status)

### 6.21.1 Detailed Description

USB-PD policy engine header file.

### 6.21.2 Function Documentation

#### 6.21.2.1 [get\\_pe\\_state\\_buf\(\)](#)

```
uint8_t* get_pe_state_buf (
 uint8_t port)
```

This function returns the Policy Engine FSM history buffer.

#### Parameters

|             |            |
|-------------|------------|
| <i>port</i> | port index |
|-------------|------------|

#### 6.21.2.2 [get\\_spec\\_rev\\_determined\(\)](#)

```
bool get_spec_rev_determined (
 uint8_t port)
```

Checks if spec rev bit in dpm\_status register is valid.

#### Parameters

|             |            |
|-------------|------------|
| <i>port</i> | port index |
|-------------|------------|

#### Returns

Returns true if spec rev is determined else return false

### 6.21.2.3 pe\_clear\_hard\_reset\_count()

```
void pe_clear_hard_reset_count (
 uint8_t port)
```

Clears hard reset count.

#### Parameters

|             |            |
|-------------|------------|
| <i>port</i> | port index |
|-------------|------------|

#### Returns

None

### 6.21.2.4 pe\_disabled()

```
void pe_disabled (
 uint8_t port)
```

Disables the policy engine. PD port is limited to receiving hard resets.

#### Parameters

|             |            |
|-------------|------------|
| <i>port</i> | port index |
|-------------|------------|

### 6.21.2.5 pe\_fsm()

```
void pe_fsm (
 uint8_t port)
```

This function runs pe state machine.

#### Parameters

|             |            |
|-------------|------------|
| <i>port</i> | port index |
|-------------|------------|

### 6.21.2.6 pe\_get\_pps\_status()

```
void pe_get_pps_status (
 uint8_t port,
 uint8_t * pps_status)
```

This function returns the PPS status.

#### Parameters

|                   |                                                     |
|-------------------|-----------------------------------------------------|
| <i>port</i>       | port index                                          |
| <i>pps_status</i> | PPS status data as defined by USB-PD specification. |

**6.21.2.7 pe\_init()**

```
void pe_init (
 uint8_t port)
```

This function initializes the policy engine. This function also initializes the pd protocol module.

**Parameters**

|             |            |
|-------------|------------|
| <i>port</i> | port index |
|-------------|------------|

**6.21.2.8 pe\_is\_busy()**

```
bool pe_is_busy (
 uint8_t port)
```

Checks if policy engine is busy in some task.

**Parameters**

|             |            |
|-------------|------------|
| <i>port</i> | port index |
|-------------|------------|

**Returns**

Returns true if busy else return false

**6.21.2.9 pe\_push\_to\_buf()**

```
void pe_push_to_buf (
 uint8_t port,
 uint8_t state)
```

This function pushes current Policy Engine state to the FSM history buffer.

**Parameters**

|              |                                          |
|--------------|------------------------------------------|
| <i>port</i>  | port index                               |
| <i>state</i> | Next state to be pushed into the buffer. |

**6.21.2.10 pe\_start()**

```
void pe_start (
 uint8_t port)
```

This function initialize the internal variables of policy engine to known state and start the pd phy.



## Parameters

|             |            |
|-------------|------------|
| <i>port</i> | port index |
|-------------|------------|

## 6.21.2.11 pe\_stop()

```
void pe_stop (
 uint8_t port)
```

Completely stops the policy engine. This will stop pd phy as well.

## Parameters

|             |            |
|-------------|------------|
| <i>port</i> | port index |
|-------------|------------|

## 6.22 pd\_common/pd\_protocol.h File Reference

```
#include <pd.h>
#include <status.h>
#include <pdss_hal.h>
#include <timer.h>
#include <srom_config.h>
```

## Data Structures

- struct [prl\\_cntrs\\_t](#)
- struct [pd\\_status\\_t](#)

## Functions

- [ccg\\_status\\_t pd\\_prot\\_init](#) (uint8\_t port, [pd\\_cbk\\_t](#) cbk)
- [ccg\\_status\\_t pd\\_prot\\_start](#) (uint8\_t port)
- [ccg\\_status\\_t pd\\_prot\\_refresh\\_roles](#) (uint8\_t port)
- [ccg\\_status\\_t pd\\_prot\\_stop](#) (uint8\_t port)
- [ccg\\_status\\_t pd\\_prot\\_rx\\_en](#) (uint8\_t port)
- [ccg\\_status\\_t pd\\_prot\\_rx\\_dis](#) (uint8\_t port, uint8\_t hard\_reset\_en)
- bool [pd\\_prot\\_is\\_busy](#) (uint8\_t port)
- [ccg\\_status\\_t pd\\_prot\\_reset\\_all](#) (uint8\_t port)
- [ccg\\_status\\_t pd\\_prot\\_reset](#) (uint8\_t port, [sop\\_t](#) sop)
- [ccg\\_status\\_t pd\\_prot\\_reset\\_rx](#) (uint8\_t port, [sop\\_t](#) sop)
- [ccg\\_status\\_t pd\\_prot\\_send\\_ctrl\\_msg](#) (uint8\_t port, [sop\\_t](#) sop, [ctrl\\_msg\\_t](#) msg\_type)
- [ccg\\_status\\_t pd\\_prot\\_send\\_data\\_msg](#) (uint8\_t port, [sop\\_t](#) sop, [data\\_msg\\_t](#) msg\_type, uint8\_t count, [pd\\_do\\_t](#) \*dobj)
- [ccg\\_status\\_t pd\\_prot\\_send\\_extd\\_\\_msg](#) (uint8\_t port, [sop\\_t](#) sop, [extd\\_msg\\_t](#) msg\_type, [pd\\_extd\\_hdr\\_t](#) ext↔\_hdr, uint8\_t \*dobj)
- [ccg\\_status\\_t pd\\_prot\\_send\\_hard\\_reset](#) (uint8\_t port)
- [ccg\\_status\\_t pd\\_prot\\_send\\_cable\\_reset](#) (uint8\_t port)
- [ccg\\_status\\_t pd\\_prot\\_en\\_bist\\_cm2](#) (uint8\_t port)

- [ccg\\_status\\_t pd\\_prot\\_dis\\_bist\\_cm2](#) (uint8\_t port)
- [ccg\\_status\\_t pd\\_prot\\_en\\_bist\\_test\\_data](#) (uint8\_t port)
- [ccg\\_status\\_t pd\\_prot\\_dis\\_bist\\_test\\_data](#) (uint8\_t port)
- [ccg\\_status\\_t pd\\_prot\\_set\\_avoid\\_retry](#) (uint8\_t port)
- [pd\\_packet\\_extd\\_t \\* pd\\_prot\\_get\\_rx\\_packet](#) (uint8\_t port)
- [ccg\\_status\\_t pd\\_prot\\_frs\\_rx\\_enable](#) (uint8\_t port)
- [ccg\\_status\\_t pd\\_prot\\_frs\\_rx\\_disable](#) (uint8\_t port)
- [ccg\\_status\\_t pd\\_prot\\_frs\\_tx\\_enable](#) (uint8\_t port)
- [ccg\\_status\\_t pd\\_prot\\_frs\\_tx\\_disable](#) (uint8\_t port)

### 6.22.1 Detailed Description

USB-PD protocol layer header file.

### 6.22.2 Function Documentation

#### 6.22.2.1 pd\_prot\_dis\_bist\_cm2()

```
ccg_status_t pd_prot_dis_bist_cm2 (
 uint8_t port)
```

This function disable sending bist carrier mode 2. There is no call back for this function This function returns after registering the request.

#### Parameters

|             |            |
|-------------|------------|
| <i>port</i> | port index |
|-------------|------------|

#### Returns

ccg\_status\_t

#### 6.22.2.2 pd\_prot\_dis\_bist\_test\_data()

```
ccg_status_t pd_prot_dis_bist_test_data (
 uint8_t port)
```

This function disables the bist test data mode.

#### Parameters

|             |            |
|-------------|------------|
| <i>port</i> | port index |
|-------------|------------|

#### Returns

ccg\_status\_t

### 6.22.2.3 pd\_prot\_en\_bist\_cm2()

```
ccg_status_t pd_prot_en_bist_cm2 (
 uint8_t port)
```

This function enables sending bist carrier mode 2. There is no call back for this function This function returns after registering the request.

#### Parameters

|             |            |
|-------------|------------|
| <i>port</i> | port index |
|-------------|------------|

#### Returns

ccg\_status\_t

### 6.22.2.4 pd\_prot\_en\_bist\_test\_data()

```
ccg_status_t pd_prot_en_bist_test_data (
 uint8_t port)
```

This function puts the receiver in bist test data mode.

#### Parameters

|             |            |
|-------------|------------|
| <i>port</i> | port index |
|-------------|------------|

#### Returns

ccg\_status\_t

### 6.22.2.5 pd\_prot\_frs\_rx\_disable()

```
ccg_status_t pd_prot_frs_rx_disable (
 uint8_t port)
```

This function disables the fast role swap receive functionality.

#### Parameters

|             |             |
|-------------|-------------|
| <i>port</i> | Port index. |
|-------------|-------------|

#### Returns

ccg\_status\_t

### 6.22.2.6 pd\_prot\_frs\_rx\_enable()

```
ccg_status_t pd_prot_frs_rx_enable (
```

```
uint8_t port)
```

This function enables the fast role swap receive functionality.

#### Parameters

|             |             |
|-------------|-------------|
| <i>port</i> | Port index. |
|-------------|-------------|

#### Returns

`ccg_status_t`

#### 6.22.2.7 pd\_prot\_frs\_tx\_disable()

```
ccg_status_t pd_prot_frs_tx_disable (
 uint8_t port)
```

This function disables the fast role swap transmit functionality.

#### Parameters

|             |             |
|-------------|-------------|
| <i>port</i> | Port index. |
|-------------|-------------|

#### Returns

`ccg_status_t`

#### 6.22.2.8 pd\_prot\_frs\_tx\_enable()

```
ccg_status_t pd_prot_frs_tx_enable (
 uint8_t port)
```

This function enables the fast role swap transmit functionality.

#### Parameters

|             |             |
|-------------|-------------|
| <i>port</i> | Port index. |
|-------------|-------------|

#### Returns

`ccg_status_t`

#### 6.22.2.9 pd\_prot\_get\_rx\_packet()

```
pd_packet_extd_t* pd_prot_get_rx_packet (
 uint8_t port)
```

This function returns pointer to received pd packet.

## Parameters

|             |            |
|-------------|------------|
| <i>port</i> | port index |
|-------------|------------|

## Returns

Returns pointer to received data if port param is not correct return null

## 6.22.2.10 pd\_prot\_init()

```
ccg_status_t pd_prot_init (
 uint8_t port,
 pd_cbk_t cbk)
```

This function sets the clock and necessary registers for PD hw ip to function and initializes the PD protocol layer.

## Parameters

|             |                           |
|-------------|---------------------------|
| <i>port</i> | port index                |
| <i>cbk</i>  | pd event handler callback |

## Returns

ccg\_status\_t

## 6.22.2.11 pd\_prot\_is\_busy()

```
bool pd_prot_is_busy (
 uint8_t port)
```

This function checks if protocol layer is busy.

## Parameters

|             |            |
|-------------|------------|
| <i>port</i> | port index |
|-------------|------------|

## Returns

Return true when busy else return false

## 6.22.2.12 pd\_prot\_refresh\_roles()

```
ccg_status_t pd_prot_refresh_roles (
 uint8_t port)
```

This function configures pd phy as per current port role /data role/contract status of port. This API does not enable the receiver.

**Parameters**

|             |            |
|-------------|------------|
| <i>port</i> | port index |
|-------------|------------|

**Returns**

`ccg_status_t`

**6.22.2.13 pd\_prot\_reset()**

```
ccg_status_t pd_prot_reset (
 uint8_t port,
 sop_t sop)
```

This function resets protocol layer(TX and RX) counter for a specific sop type.

**Parameters**

|             |            |
|-------------|------------|
| <i>port</i> | port index |
| <i>sop</i>  | sop type   |

**Returns**

`ccg_status_t`

**6.22.2.14 pd\_prot\_reset\_all()**

```
ccg_status_t pd_prot_reset_all (
 uint8_t port)
```

This function resets the protocol layer(TX and RX) counters for each sop type.

**Parameters**

|             |            |
|-------------|------------|
| <i>port</i> | port index |
|-------------|------------|

**Returns**

`ccg_status_t`

**6.22.2.15 pd\_prot\_reset\_rx()**

```
ccg_status_t pd_prot_reset_rx (
 uint8_t port,
 sop_t sop)
```

This function resets protocol layer RX only counter for a specific sop type.

## Parameters

|             |            |
|-------------|------------|
| <i>port</i> | port index |
| <i>sop</i>  | sop type   |

## Returns

ccg\_status\_t

## 6.22.2.16 pd\_prot\_rx\_dis()

```
ccg_status_t pd_prot_rx_dis (
 uint8_t port,
 uint8_t hard_reset_en)
```

This function disables the bmc receiver.

## Parameters

|                      |                                                                                       |
|----------------------|---------------------------------------------------------------------------------------|
| <i>port</i>          | port index                                                                            |
| <i>hard_reset_en</i> | When 0 means rx is completely disabled, When 1 means only hard reset can be received. |

## Returns

ccg\_status\_t

## 6.22.2.17 pd\_prot\_rx\_en()

```
ccg_status_t pd_prot_rx_en (
 uint8_t port)
```

This function enables the bmc receiver.

## Parameters

|             |            |
|-------------|------------|
| <i>port</i> | port index |
|-------------|------------|

## Returns

ccg\_status\_t

## 6.22.2.18 pd\_prot\_send\_cable\_reset()

```
ccg_status_t pd_prot_send_cable_reset (
 uint8_t port)
```

This function sends a cable reset. Results will be known to caller via callback function registered in [pd\\_phy\\_init\(\)](#) if this function returns success. This function returns after registering the request.

## Parameters

|             |            |
|-------------|------------|
| <i>port</i> | port index |
|-------------|------------|

## Returns

`ccg_status_t`

6.22.2.19 `pd_prot_send_ctrl_msg()`

```
ccg_status_t pd_prot_send_ctrl_msg (
 uint8_t port,
 sop_t sop,
 ctrl_msg_t msg_type)
```

This function sends a control message. Results will be known to caller via callback function registered in `pd_prot_init()` if this function returns success. This function returns after registering the request.

## Parameters

|                 |                       |
|-----------------|-----------------------|
| <i>port</i>     | port index            |
| <i>sop</i>      | sop type              |
| <i>msg_type</i> | control message type. |

## Returns

`ccg_status_t`

6.22.2.20 `pd_prot_send_data_msg()`

```
ccg_status_t pd_prot_send_data_msg (
 uint8_t port,
 sop_t sop,
 data_msg_t msg_type,
 uint8_t count,
 pd_do_t * dobj)
```

This function sends a data message. Results will be known to caller via callback function registered in `pd_phy_init()` if this function returns success. This function returns after registering the request.

## Parameters

|                 |                         |
|-----------------|-------------------------|
| <i>port</i>     | port index              |
| <i>sop</i>      | sop type                |
| <i>msg_type</i> | data message type       |
| <i>count</i>    | data objects count      |
| <i>dobj</i>     | pointer to data objects |



## Returns

ccg\_status\_t

## 6.22.2.21 pd\_prot\_send\_extd\_msg()

```
ccg_status_t pd_prot_send_extd_msg (
 uint8_t port,
 sop_t sop,
 extd_msg_t msg_type,
 pd_extd_hdr_t ext_hdr,
 uint8_t * dobj)
```

This function sends an extended message. Results will be known to caller via callback function registered in [pd\\_phy\\_init\(\)](#) if this function returns success. This function returns after registering the request.

## Parameters

|                 |                        |
|-----------------|------------------------|
| <i>port</i>     | port index             |
| <i>sop</i>      | sop type               |
| <i>msg_type</i> | data message type      |
| <i>ext_hdr</i>  | 16 bit extended header |
| <i>dobj</i>     | pointer to data        |

## Returns

ccg\_status\_t

## 6.22.2.22 pd\_prot\_send\_hard\_reset()

```
ccg_status_t pd_prot_send_hard_reset (
 uint8_t port)
```

This function sends a hard reset. Results will be known to caller via callback function registered in [pd\\_phy\\_init\(\)](#) if this function returns success. This function returns after registering the request.

## Parameters

|             |            |
|-------------|------------|
| <i>port</i> | port index |
|-------------|------------|

## Returns

ccg\_status\_t

## 6.22.2.23 pd\_prot\_set\_avoid\_retry()

```
ccg_status_t pd_prot_set_avoid_retry (
 uint8_t port)
```

This function can be used by higher layers to avoid retry on CRC expire for a particular message. When this flag is set before calling `pd_prot_send_data_msg()` or `pd_send_crtl_msg()` APIs, retry is avoided on CRC failure. This is one time only. Flag is automatically cleared after transmission(success or fail).

#### Parameters

|             |            |
|-------------|------------|
| <i>port</i> | port index |
|-------------|------------|

#### Returns

`ccg_status_t`

#### 6.22.2.24 pd\_prot\_start()

```
ccg_status_t pd_prot_start (
 uint8_t port)
```

This function starts the protocol layer and configures pd phy as per current port role /data role/contract status of port. This API does not enable the receiver.

#### Parameters

|             |            |
|-------------|------------|
| <i>port</i> | port index |
|-------------|------------|

#### Returns

`ccg_status_t`

#### 6.22.2.25 pd\_prot\_stop()

```
ccg_status_t pd_prot_stop (
 uint8_t port)
```

This function completely stops the pd hw and put it in lowest power state.

#### Parameters

|             |            |
|-------------|------------|
| <i>port</i> | port index |
|-------------|------------|

#### Returns

`ccg_status_t`

## 6.23 pd\_common/typec\_manager.h File Reference

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

## Functions

- void [typec\\_init](#) (uint8\_t port)
- void [typec\\_start](#) (uint8\_t port)
- void [typec\\_stop](#) (uint8\_t port)
- bool [typec\\_is\\_busy](#) (uint8\_t port)
- void [typec\\_sync\\_toggle](#) (void)
- void [typec\\_deepsleep](#) (uint8\_t port)
- void [typec\\_wakeup](#) (void)
- void [typec\\_fsm](#) (uint8\_t port)
- void [typec\\_assert\\_rp](#) (uint8\_t port, uint8\_t channel)
- void [typec\\_assert\\_rd](#) (uint8\_t port, uint8\_t channel)
- void [typec\\_change\\_rp](#) (uint8\_t port, [rp\\_term\\_t](#) rp)

### 6.23.1 Detailed Description

Type-C manager header file.

### 6.23.2 Function Documentation

#### 6.23.2.1 typec\_assert\_rd()

```
void typec_assert_rd (
 uint8_t port,
 uint8_t channel)
```

This function asserts Rd and deasserts Rp for the specified CC line.

##### Parameters

|                |             |
|----------------|-------------|
| <i>port</i>    | Port index. |
| <i>channel</i> | CC line.    |

#### 6.23.2.2 typec\_assert\_rp()

```
void typec_assert_rp (
 uint8_t port,
 uint8_t channel)
```

This function asserts Rp and deasserts Rd for the specified CC line.

##### Parameters

|                |             |
|----------------|-------------|
| <i>port</i>    | Port index. |
| <i>channel</i> | CC line.    |

### 6.23.2.3 typec\_change\_rp()

```
void typec_change_rp (
 uint8_t port,
 rp_term_t rp)
```

This function changes Rp. If port is in connected state Rp will be changed only on active channel. If port is not connected then Rp gets updated on both CC channels.

#### Parameters

|             |                   |
|-------------|-------------------|
| <i>port</i> | Port index.       |
| <i>rp</i>   | Desired Rp value. |

### 6.23.2.4 typec\_deepsleep()

```
void typec_deepsleep (
 uint8_t port)
```

This function configures Type C functionality in the PD block when entering deepsleep.

#### Parameters

|             |             |
|-------------|-------------|
| <i>port</i> | Port index. |
|-------------|-------------|

### 6.23.2.5 typec\_fsm()

```
void typec_fsm (
 uint8_t port)
```

This function implements the Type C state machine.

#### Parameters

|             |             |
|-------------|-------------|
| <i>port</i> | Port index. |
|-------------|-------------|

### 6.23.2.6 typec\_init()

```
void typec_init (
 uint8_t port)
```

This function initializes the Type C manager. This function must be called once during system init.

#### Parameters

|             |             |
|-------------|-------------|
| <i>port</i> | Port index. |
|-------------|-------------|

### 6.23.2.7 typec\_is\_busy()

```
bool typec_is_busy (
 uint8_t port)
```

This function checks if the Type C FSM is busy.

#### Parameters

|             |             |
|-------------|-------------|
| <i>port</i> | Port index. |
|-------------|-------------|

#### Returns

Returns true if busy, false otherwise.

### 6.23.2.8 typec\_start()

```
void typec_start (
 uint8_t port)
```

This function starts the Type C functionality in the PD hardware block.

#### Parameters

|             |             |
|-------------|-------------|
| <i>port</i> | Port index. |
|-------------|-------------|

### 6.23.2.9 typec\_stop()

```
void typec_stop (
 uint8_t port)
```

This function completely disables the Type C FSM and the Type C functionality in the PD hardware block.

#### Parameters

|             |             |
|-------------|-------------|
| <i>port</i> | Port index. |
|-------------|-------------|

## 6.24 pd\_hal/hal\_ccgx.h File Reference

```
#include "config.h"
#include "pd.h"
#include "stdint.h"
#include "stdbool.h"
```

### Typedefs

- typedef void(\* [vbus\\_ocp\\_cbk\\_t](#)) (uint8\_t port)

## Functions

- void [system\\_init](#) (void)
- uint8\_t [system\\_vbus\\_ocp\\_en](#) (uint8\_t port, uint32\_t cur, [vbus\\_ocp\\_cbk\\_t](#) cbk)
- uint8\_t [system\\_vbus\\_ocp\\_dis](#) (uint8\_t port)
- void [vbus\\_ocp\\_handler](#) (uint8\_t port)
- void [system\\_disconnect\\_ovp\\_trip](#) (uint8\_t port)
- void [system\\_connect\\_ovp\\_trip](#) (uint8\_t port)
- uint8\_t [system\\_vbus\\_scp\\_en](#) (uint8\_t port, uint32\_t cur, [vbus\\_ocp\\_cbk\\_t](#) cbk)
- uint8\_t [system\\_vbus\\_scp\\_dis](#) (uint8\_t port)
- void [vbus\\_scp\\_handler](#) (uint8\_t port)
- uint8\_t [system\\_vbus\\_rcp\\_en](#) (uint8\_t port, [vbus\\_ocp\\_cbk\\_t](#) cbk)
- uint8\_t [system\\_vbus\\_rcp\\_dis](#) (uint8\_t port)
- void [vbus\\_rcp\\_handler](#) (uint8\_t port)

### 6.24.1 Detailed Description

PD and Type-C HAL layer for CCGx device family.

### 6.24.2 Function Documentation

#### 6.24.2.1 [system\\_connect\\_ovp\\_trip\(\)](#)

```
void system_connect_ovp_trip (
 uint8_t port)
```

This function connects the OVP comparator output to OVP Trip GPIO.

#### Parameters

|             |             |
|-------------|-------------|
| <i>port</i> | USB-PD port |
|-------------|-------------|

#### Returns

None

**Applicable devices:** CCG4

#### 6.24.2.2 [system\\_disconnect\\_ovp\\_trip\(\)](#)

```
void system_disconnect_ovp_trip (
 uint8_t port)
```

This function disconnects the OVP comparator output from OVP Trip GPIO. Before disconnecting, this function also make sure OVP Trip GPIO is strongly driving the last output of comparator.

#### Parameters

|             |             |
|-------------|-------------|
| <i>port</i> | USB-PD port |
|-------------|-------------|

**Returns**

None

**Applicable devices:** CCG4**6.24.2.3 system\_init()**

```
void system_init (
 void)
```

USB-PD system initialization.

This function initializes the PD block clocks by setting the divider values for PERI registers and enabling the corresponding control registers.

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6.**6.24.2.4 system\_vbus\_ocp\_dis()**

```
uint8_t system_vbus_ocp_dis (
 uint8_t port)
```

Disables VBus OCP checks on the specified port.

**Parameters**

|             |                                                  |
|-------------|--------------------------------------------------|
| <i>port</i> | USB-PD port on which to disable VBus OCP checks. |
|-------------|--------------------------------------------------|

**Returns**

Returns 1 if params are ok else return 0

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6.**6.24.2.5 system\_vbus\_ocp\_en()**

```
uint8_t system_vbus_ocp_en (
 uint8_t port,
 uint32_t cur,
 vbus_ocp_cbk_t cbk)
```

Enables VBus OCP checks on the specified port.

**Parameters**

|             |                                                   |
|-------------|---------------------------------------------------|
| <i>port</i> | USB-PD port on which to enable VBus OCP checks.   |
| <i>cur</i>  | Maximum current (load) expected in 10 mA units.   |
| <i>cbk</i>  | Function to be called when OCP event is detected. |

**Returns**

Returns 1 if params are ok else return 0

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6.

### 6.24.2.6 system\_vbus\_rcp\_dis()

```
uint8_t system_vbus_rcp_dis (
 uint8_t port)
```

Disables VBus RCP checks on the specified port.

#### Parameters

|             |                                                  |
|-------------|--------------------------------------------------|
| <i>port</i> | USB-PD port on which to disable VBus RCP checks. |
|-------------|--------------------------------------------------|

#### Returns

Returns 1 if params are ok else return 0

**Applicable devices:** CCG6.

### 6.24.2.7 system\_vbus\_rcp\_en()

```
uint8_t system_vbus_rcp_en (
 uint8_t port,
 vbus_ocp_cbk_t cbk)
```

Enables VBus RCP checks on the specified port.

#### Parameters

|             |                                                   |
|-------------|---------------------------------------------------|
| <i>port</i> | USB-PD port on which to enable VBus RCP checks.   |
| <i>cbk</i>  | Function to be called when RCP event is detected. |

#### Returns

Returns 1 if params are ok else return 0

**Applicable devices:** CCG6.

### 6.24.2.8 system\_vbus\_scp\_dis()

```
uint8_t system_vbus_scp_dis (
 uint8_t port)
```

Disables VBus SCP checks on the specified port.

#### Parameters

|             |                                                  |
|-------------|--------------------------------------------------|
| <i>port</i> | USB-PD port on which to disable VBus SCP checks. |
|-------------|--------------------------------------------------|

#### Returns

Returns 1 if params are ok else return 0

**Applicable devices:** CCG3PA, CCG3PA2, CCG6.



## 6.24.2.9 system\_vbus\_scp\_en()

```
uint8_t system_vbus_scp_en (
 uint8_t port,
 uint32_t cur,
 vbus_ocp_cbk_t cbk)
```

Enables VBus SCP checks on the specified port.

## Parameters

|             |                                                          |
|-------------|----------------------------------------------------------|
| <i>port</i> | USB-PD port on which to enable VBus SCP checks.          |
| <i>cur</i>  | Current limit for Short-Circuit detection in 10mA units. |
| <i>cbk</i>  | Function to be called when SCP event is detected.        |

## Returns

Returns 1 if params are ok else return 0

**Applicable devices:** CCG3PA, CCG3PA2, CCG6.

## 6.24.2.10 vbus\_ocp\_handler()

```
void vbus_ocp_handler (
 uint8_t port)
```

Vbus OCP interrupt handler.

## Parameters

|             |                                                       |
|-------------|-------------------------------------------------------|
| <i>port</i> | USB-PD port on which the OCP interrupt was triggered. |
|-------------|-------------------------------------------------------|

## Returns

None

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6.

## 6.24.2.11 vbus\_rcp\_handler()

```
void vbus_rcp_handler (
 uint8_t port)
```

Vbus RCP interrupt handler.

## Parameters

|             |                                                       |
|-------------|-------------------------------------------------------|
| <i>port</i> | USB-PD port on which the RCP interrupt was triggered. |
|-------------|-------------------------------------------------------|

## Returns

None

**Applicable devices:** CCG6.

### 6.24.2.12 vbus\_scp\_handler()

```
void vbus_scp_handler (
 uint8_t port)
```

Vbus SCP interrupt handler.

#### Parameters

|             |                                                       |
|-------------|-------------------------------------------------------|
| <i>port</i> | USB-PD port on which the SCP interrupt was triggered. |
|-------------|-------------------------------------------------------|

#### Returns

None

**Applicable devices:** CCG3PA, CCG3PA2, CCG6.

## 6.25 pd\_hal/hpd.h File Reference

```
#include <stdint.h>
#include <stdbool.h>
#include <ccgx_regs.h>
#include <status.h>
```

### Macros

- `#define HPD_IP_DIRECTION (0)`
- `#define HPD_OP_DIRECTION (PDSS_CTRL_HPD_DIRECTION)`
- `#define HPD_EVENT_MASK (3u)`
- `#define HPD_EVENT_0_POS (0)`
- `#define HPD_EVENT_1_POS (2)`
- `#define HPD_EVENT_2_POS (4)`
- `#define HPD_EVENT_3_POS (6)`
- `#define HPD_GET_EVENT_0(hpd_queue) (((hpd_queue) >> HPD_EVENT_0_POS) & HPD_EVENT_MASK)`
- `#define HPD_GET_EVENT_1(hpd_queue) (((hpd_queue) >> HPD_EVENT_1_POS) & HPD_EVENT_MASK)`
- `#define HPD_GET_EVENT_2(hpd_queue) (((hpd_queue) >> HPD_EVENT_2_POS) & HPD_EVENT_MASK)`
- `#define HPD_GET_EVENT_3(hpd_queue) (((hpd_queue) >> HPD_EVENT_3_POS) & HPD_EVENT_MASK)`

### Typedefs

- `typedef void(* hpd_event_cbk_t) (uint8_t port, hpd_event_type_t event)`

### Enumerations

- `enum hpd_event_type_t {`  
`HPD_EVENT_NONE = 0, HPD_EVENT_UNPLUG, HPD_EVENT_PLUG, HPD_EVENT_IRQ,`  
`HPD_COMMAND_DONE }`

## Functions

- [ccg\\_status\\_t hpd\\_receive\\_init](#) (uint8\_t port, [hpd\\_event\\_cbk\\_t](#) cbk)
- [ccg\\_status\\_t hpd\\_transmit\\_init](#) (uint8\_t port, [hpd\\_event\\_cbk\\_t](#) cbk)
- [ccg\\_status\\_t hpd\\_deinit](#) (uint8\_t port)
- [bool hpd\\_receive\\_get\\_status](#) (uint8\_t port)
- [ccg\\_status\\_t hpd\\_transmit\\_sendevt](#) (uint8\_t port, [hpd\\_event\\_type\\_t](#) evtype, bool wait)
- [void hpd\\_sleep\\_entry](#) (uint8\_t port)
- [void hpd\\_wakeup](#) (uint8\_t port, bool value)
- [void hpd\\_rx\\_sleep\\_entry](#) (uint8\_t port, bool hpd\_state)
- [void hpd\\_rx\\_wakeup](#) (uint8\_t port)
- [bool is\\_hpd\\_rx\\_state\\_idle](#) (uint8\_t port)

### 6.25.1 Detailed Description

Hotplug Detect (HPD) driver header file.

### 6.25.2 Macro Definition Documentation

#### 6.25.2.1 HPD\_EVENT\_0\_POS

```
#define HPD_EVENT_0_POS (0)
```

HPD Event 0 bit position.

#### 6.25.2.2 HPD\_EVENT\_1\_POS

```
#define HPD_EVENT_1_POS (2)
```

HPD Event 1 bit position.

#### 6.25.2.3 HPD\_EVENT\_2\_POS

```
#define HPD_EVENT_2_POS (4)
```

HPD Event 2 bit position.

#### 6.25.2.4 HPD\_EVENT\_3\_POS

```
#define HPD_EVENT_3_POS (6)
```

HPD Event 3 bit position.

#### 6.25.2.5 HPD\_EVENT\_MASK

```
#define HPD_EVENT_MASK (3u)
```

HPD HW register event mask.

### 6.25.2.6 HPD\_GET\_EVENT\_0

```
#define HPD_GET_EVENT_0(
 hpd_queue) ((hpd_queue) >> HPD_EVENT_0_POS) & HPD_EVENT_MASK)
```

Get the first HPD event from queue status.

### 6.25.2.7 HPD\_GET\_EVENT\_1

```
#define HPD_GET_EVENT_1(
 hpd_queue) ((hpd_queue) >> HPD_EVENT_1_POS) & HPD_EVENT_MASK)
```

Get the second HPD event from queue status.

### 6.25.2.8 HPD\_GET\_EVENT\_2

```
#define HPD_GET_EVENT_2(
 hpd_queue) ((hpd_queue) >> HPD_EVENT_2_POS) & HPD_EVENT_MASK)
```

Get the third HPD event from queue status.

### 6.25.2.9 HPD\_GET\_EVENT\_3

```
#define HPD_GET_EVENT_3(
 hpd_queue) ((hpd_queue) >> HPD_EVENT_3_POS) & HPD_EVENT_MASK)
```

Get the fourth HPD event from queue status.

### 6.25.2.10 HPD\_IP\_DIRECTION

```
#define HPD_IP_DIRECTION (0)
```

HPD Input direction is used when sensing HPD output from a monitor. This is used in applications like video dongles and monitors.

### 6.25.2.11 HPD\_OP\_DIRECTION

```
#define HPD_OP_DIRECTION (PDSS_CTRL_HPD_DIRECTION)
```

HPD Output direction is used when driving HPD output to a DP source. This is used in applications like PD port controllers in PC/Mobile platforms.

## 6.25.3 Enumeration Type Documentation

### 6.25.3.1 hpd\_event\_type\_t

```
enum hpd_event_type_t
```

List of HPD events detected by USBPD block. The UNPLUG, PLUG and IRQ events are used in the case of a DisplayPort Sink implementation, and the COMMAND\_DONE event is used in the case of a DP Source implementation.

#### Enumerator

|                |           |
|----------------|-----------|
| HPD_EVENT_NONE | No event. |
|----------------|-----------|

## Enumerator

|                  |                                    |
|------------------|------------------------------------|
| HPD_EVENT_UNPLUG | DP Unplug event.                   |
| HPD_EVENT_PLUG   | DP Plug event.                     |
| HPD_EVENT_IRQ    | DP IRQ event.                      |
| HPD_COMMAND_DONE | Requested HPD command is complete. |

## 6.25.4 Function Documentation

## 6.25.4.1 hpd\_deinit()

```
ccg_status_t hpd_deinit (
 uint8_t port)
```

Disable the HPD functionality on the specified PD port. This is used for HPD receive as well transmit de-initialization.

## Parameters

|             |                                                                   |
|-------------|-------------------------------------------------------------------|
| <i>port</i> | PD port index. Caller should ensure to provide only valid values. |
|-------------|-------------------------------------------------------------------|

## Returns

Return CCG\_STAT\_SUCCESS in case of success, error code otherwise.

**Applicable devices:** CCG3, CCG4, CCG5, CCG5C, CCG6.

## 6.25.4.2 hpd\_receive\_get\_status()

```
bool hpd_receive_get_status (
 uint8_t port)
```

Returns the current status of the HPD input signal. This function can only be used if the HPD receive block has been enabled using the hpd\_receive\_init function.

## Parameters

|             |                |
|-------------|----------------|
| <i>port</i> | pd port index. |
|-------------|----------------|

## Returns

Current HPD input signal status.

**Applicable devices:** CCG3, CCG4, CCG5, CCG5C, CCG6.

## 6.25.4.3 hpd\_receive\_init()

```
ccg_status_t hpd_receive_init (
 uint8_t port,
 hpd_event_cbk_t cbk)
```

Enable the HPD-Receive functionality for the specified PD port. This function shall be used in DP Sink designs and should not be combined with the `hpd_transmit_init` call.

#### Parameters

|             |                                                                   |
|-------------|-------------------------------------------------------------------|
| <i>port</i> | PD port index. Caller should ensure to provide only valid values. |
| <i>cbk</i>  | callback function to be called when there is an HPD event.        |

#### Returns

Returns `CCG_STAT_SUCCESS` in case of success, error code otherwise.

**Applicable devices:** CCG3, CCG4, CCG5, CCG5C, CCG6.

#### 6.25.4.4 `hpd_rx_sleep_entry()`

```
void hpd_rx_sleep_entry (
 uint8_t port,
 bool hpd_state)
```

Prepare the HPD RX block for deep-sleep.

This routine implements a firmware workaround to retain HPD signal state before entering deep sleep. All details of the workaround are captured in function definition.

#### Parameters

|                  |                                                                                 |
|------------------|---------------------------------------------------------------------------------|
| <i>port</i>      | Port whose HPD RX is active. Caller should ensure to provide only valid values. |
| <i>hpd_state</i> | Connection status of HPD                                                        |

#### Returns

None

**Applicable devices:** CCG3, CCG4, CCG5, CCG5C, CCG6.

#### 6.25.4.5 `hpd_rx_wakeup()`

```
void hpd_rx_wakeup (
 uint8_t port)
```

Restore the HPD RX block function after CCG wakes up from deep sleep. Implements the wakeup portion of the work-around to retail HPD signal state.

#### Parameters

|             |                                                        |
|-------------|--------------------------------------------------------|
| <i>port</i> | Port whose HPD RX module settings have to be restored. |
|-------------|--------------------------------------------------------|

#### Returns

None

**Applicable devices:** CCG3, CCG4, CCG5, CCG5C, CCG6.

## 6.25.4.6 hpd\_sleep\_entry()

```
void hpd_sleep_entry (
 uint8_t port)
```

Prepare the HPD transmit block for deep-sleep. This function changes the IO mode of the HPD signal to GPIO, so as to avoid glitches on the signal during the active - sleep - active power mode transitions of the CCG device.

## Parameters

|             |                                         |
|-------------|-----------------------------------------|
| <i>port</i> | Port whose HPD output is to be updated. |
|-------------|-----------------------------------------|

## Returns

None

**Applicable devices:** CCG3, CCG4, CCG5, CCG5C, CCG6.

## 6.25.4.7 hpd\_transmit\_init()

```
ccg_status_t hpd_transmit_init (
 uint8_t port,
 hpd_event_cbk_t cbk)
```

Enable the HPD-Transmit functionality for the specified PD port. This function shall be used in DP source designs and should not be combined with the hpd\_receive\_init call.

## Parameters

|             |                                                                   |
|-------------|-------------------------------------------------------------------|
| <i>port</i> | PD port index. Caller should ensure to provide only valid values. |
| <i>cbk</i>  | callback to be used for command completion event.                 |

## Returns

Returns CCG\_STAT\_SUCCESS in case of success, error code otherwise.

**Applicable devices:** CCG3, CCG4, CCG5, CCG5C, CCG6.

## 6.25.4.8 hpd\_transmit\_sendevt()

```
ccg_status_t hpd_transmit_sendevt (
 uint8_t port,
 hpd_event_type_t evtype,
 bool wait)
```

Send the desired HPD event out through the HPD GPIO. Only the HPD\_EVENT\_UNPLUG, HPD\_EVENT\_UNPLUG and HPD\_EVENT\_IRQ events should be requested.

## Parameters

|               |                                                         |
|---------------|---------------------------------------------------------|
| <i>port</i>   | Port on which HPD event is to be sent.                  |
| <i>evtype</i> | Type of HPD event to be sent.                           |
| <i>wait</i>   | Whether function should wait until command is complete. |

**Returns**

Returns CCG\_STAT\_SUCCESS in case of success, error code otherwise.

**Applicable devices:** CCG3, CCG4, CCG5, CCG5C, CCG6.

**6.25.4.9 hpd\_wakeup()**

```
void hpd_wakeup (
 uint8_t port,
 bool value)
```

Restore the HPD Tx signal driver after CCG wakes from deep-sleep.

**Parameters**

|              |                                                                                                                                               |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| <i>port</i>  | Port whose HPD signal is to be restored.                                                                                                      |
| <i>value</i> | Startup value for the HPD signal. The APP layer is expected to identify the desired (PLUG/UNPLUG) state of the signal and pass it to the HAL. |

**Returns**

None

**Applicable devices:** CCG3, CCG4, CCG5, CCG5C, CCG6.

**6.25.4.10 is\_hpd\_rx\_state\_idle()**

```
bool is_hpd_rx_state_idle (
 uint8_t port)
```

Returns the state of HPD RX activity timer. This check should be used to prevent device entry into a low power sleep state while an HPD transition is in progress.

**Parameters**

|             |                                           |
|-------------|-------------------------------------------|
| <i>port</i> | Port whose HPD RX state is to be checked. |
|-------------|-------------------------------------------|

**Returns**

Status of timer. true if timer is active, false otherwise.

**Applicable devices:** CCG3, CCG4, CCG5, CCG5C, CCG6.

**6.26 pd\_hal/pdss\_hal.h File Reference**

```
#include <ccgx_regs.h>
#include <pd.h>
#include <status.h>
```

**Data Structures**

- struct [pasc\\_valley\\_table\\_t](#)



## Macros

- #define PDSS\_DRP\_TOGGLE\_PERIOD\_MS (75u)
- #define PDSS\_DRP\_HIGH\_PERIOD\_MS (30u)
- #define PDSS\_DRP\_TOGGLE\_PERIOD\_VAL (2500u)
- #define PDSS\_DRP\_HIGH\_PERIOD\_VAL (1100u)
- #define PDSS\_CC\_FILT\_CYCLES (10)
- #define PGDO\_PD\_ISINK\_TERMINAL\_VAL (0x3F)
- #define PGDO\_PD\_ISINK\_TERMINAL\_VAL\_1 (0x1F)
- #define AUTO\_CTRL\_MESSAGE\_GOODCRC\_MASK\_CFG (0xFFFFFFFFDu)
- #define AUTO\_DATA\_MESSAGE\_GOODCRC\_MASK\_CFG (0xFFFFFFFFFu)
- #define AUTO\_EXTD\_MESSAGE\_GOODCRC\_MASK\_CFG (0xFFFFFFFFFu)
- #define PD\_MSG\_HDR\_REV2\_IGNORE\_MASK (0x8010)
- #define EXPECTED\_GOOD\_CRC\_HDR\_MASK (0x7E0Fu)
- #define EXPECTED\_GOOD\_CRC\_HDR\_DIFF\_MASK\_REV3 (PD\_MSG\_HDR\_REV2\_IGNORE\_MASK)
- #define EXPECTED\_GOOD\_CRC\_CLEAR\_MASK (0xF1FFu)
- #define RX\_CNT\_MAX\_VAL (0x0Fu)
- #define RX\_UI\_BOUNDARY\_DELTA\_VAL (0x02u)
- #define PDSS\_MAX\_TX\_MEM\_SIZE (16u)
- #define PDSS\_MAX\_TX\_MEM\_HALF\_SIZE (8u)
- #define PDSS\_MAX\_RX\_MEM\_SIZE (16u)
- #define PDSS\_MAX\_RX\_MEM\_HALF\_SIZE (8u)
- #define VBUS\_OCP\_MODE\_EXT (0u)
- #define VBUS\_OCP\_MODE\_INT (1u)
- #define VBUS\_OCP\_MODE\_INT\_AUTOCTRL (2u)
- #define VBUS\_OCP\_MODE\_INT\_SW\_DB (3u)
- #define VBUS\_OCP\_MODE\_POLLING (4u)
- #define VBUS\_OCP\_GPIO\_ACTIVE\_LOW (0u)
- #define VBUS\_OCP\_GPIO\_ACTIVE\_HIGH (1u)

## Typedefs

- typedef void(\* PD\_ADC\_CB\_T) (uint8\_t port, bool comp\_out)
- typedef void(\* pd\_phy\_cbk\_t) (uint8\_t port, pd\_phy\_evt\_t event)
- typedef void(\* vbus\_cf\_cbk\_t) (uint8\_t port, bool cf\_state)
- typedef void(\* pd\_cmp\_cbk\_t) (uint8\_t port, bool state)
- typedef void(\* pd\_supply\_change\_cbk\_t) (uint8\_t port, ccg\_supply\_t supply\_id, bool present)
- typedef void(\* pasc\_valley\_cbk\_t) (uint8\_t port, ccg\_status\_t status)
- typedef void(\* vbus\_load\_chg\_cbk\_t) (uint8\_t port)
- typedef void(\* pasc\_ptdrv\_cont\_cbk\_t) (uint8\_t port)

## Enumerations

- enum PD\_ADC\_VREF\_T { PD\_ADC\_VREF\_PROG = 0, PD\_ADC\_VREF\_VDDD }
- enum PD\_ADC\_ID\_T { PD\_ADC\_ID\_0, PD\_ADC\_ID\_1, PD\_ADC\_NUM\_ADC }
- enum PD\_ADC\_INPUT\_T { PD\_ADC\_INPUT\_AMUX\_A, PD\_ADC\_INPUT\_AMUX\_B, PD\_ADC\_INPUT\_BANDGAP, PD\_ADC\_INPUT\_BJT, PD\_ADC\_NUM\_INPUT }
- enum PD\_ADC\_INT\_T { PD\_ADC\_INT\_DISABLED, PD\_ADC\_INT\_FALLING, PD\_ADC\_INT\_RISING, PD\_ADC\_INT\_BOTH }

- enum `pd_phy_evt_t` {  
`PD_PHY_EVT_TX_MSG_COLLISION`, `PD_PHY_EVT_TX_MSG_PHY_IDLE`, `PD_PHY_EVT_TX_MSG_FAILED`,  
`PD_PHY_EVT_TX_MSG_SUCCESS`,  
`PD_PHY_EVT_TX_RST_COLLISION`, `PD_PHY_EVT_TX_RST_SUCCESS`, `PD_PHY_EVT_RX_MSG`,  
`PD_PHY_EVT_RX_MSG_CMPLT`,  
`PD_PHY_EVT_RX_RST`, `PD_PHY_EVT_FRS_SIG_RCVD`, `PD_PHY_EVT_FRS_SIG_SENT`, `PD_PHY_EVT_CRC_ERROR`  
}
- enum `vbus_ovp_mode_t` { `VBUS_OVP_MODE_ADC`, `VBUS_OVP_MODE_UVOV`, `VBUS_OVP_MODE_UVOV_AUTOCTRL` }
- enum `vbus_uvp_mode_t` { `VBUS_UVP_MODE_ADC`, `VBUS_UVP_MODE_INT_COMP`, `VBUS_UVP_MODE_INT_COMP_AUTO` }
- enum `frs_tx_source_t` { `FRS_TX_SOURCE_CPU` = 0, `FRS_TX_SOURCE_GPIO`, `FRS_TX_SOURCE_ADC1`,  
`FRS_TX_SOURCE_ADC2` }
- enum `pd_fet_dr_t` { `PD_FET_DR_ACTIVE_HIGH` = 0, `PD_FET_DR_ACTIVE_LOW`, `PD_FET_DR_N_JN_FET`,  
`PD_FET_DR_P_JN_FET` }
- enum `sbu_switch_state_t` {  
`SBU_NOT_CONNECTED`, `SBU_CONNECT_AUX1`, `SBU_CONNECT_AUX2`, `SBU_CONNECT_LSTX`,  
`SBU_CONNECT_LSRX`, `SBU_MAX_STATE` }
- enum `aux_resistor_config_t` {  
`AUX_NO_RESISTOR` = 0, `AUX_1_1MEG_PU_RESISTOR` = 7, `AUX_1_100K_PD_RESISTOR` = 8,  
`AUX_1_470K_PD_RESISTOR` = 9,  
`AUX_2_100K_PU_RESISTOR` = 10, `AUX_2_4P7MEG_PD_RESISTOR` = 11, `AUX_2_1MEG_PD_RESISTOR`  
= 12, `AUX_MAX_RESISTOR_CONFIG` }
- enum `lscsa_app_config_t` {  
`LSCSA_OCP_CONFIG`, `LSCSA_EA_CONFIG`, `LSCSA_PFC_OFF_CONFIG`, `LSCSA_PFC_ON_CONFIG`,  
`LSCSA_SR_OFF_CONFIG`, `LSCSA_SR_ON_CONFIG`, `LSCSA_MAX_CONFIG_VALUE` }
- enum `comp_id_t` {  
`COMP_ID_UV` = 0, `COMP_ID_OV` = 1, `COMP_ID_VBUS_C_MON` = 2, `COMP_ID_VBUS_MON` = 2,  
`COMP_ID_VBUS_DISCHARGE` = 3, `COMP_ID_VSYS_DET` = 3, `COMP_ID_LSCSA_SCP` = 4,  
`COMP_ID_DP_DETACH` = 4,  
`COMP_ID_DM_DETACH` = 5, `COMP_ID_LSCSA_OCP` = 4, `COMP_ID_LSCSA_PFC` = 5, `COMP_ID_VSRC_NEW_P`  
= 6,  
`COMP_ID_VSRC_NEW_M` = 7, `COMP_ID_MAX` = 9 }
- enum `comp_tr_id_t` { `COMP_TR_ID_SR` = 0, `COMP_TR_ID_MAX` = 1 }
- enum `filter_id_t` {  
`FILTER_ID_UV` = 0, `FILTER_ID_OV` = 1, `FILTER_ID_DISCH_EN` = 2, `FILTER_ID_LSCSA_OCP` = 3,  
`FILTER_ID_LSCSA_PFC` = 4, `FILTER_ID_LSCSA_SR` = 5, `FILTER_ID_VSRC_NEW_P` = 8, `FILTER_ID_VSRC_NEW_M`  
= 9,  
`FILTER_ID_PDS_SCP` = 10, `FILTER_ID_MAX` }
- enum `filter_edge_detect_cfg_t` {  
`FILTER_CFG_POS_DIS_NEG_DIS`, `FILTER_CFG_POS_DIS_NEG_EN`, `FILTER_CFG_POS_EN_NEG_DIS`,  
`FILTER_CFG_POS_EN_NEG_EN`,  
`FILTER_CFG_MAX` }
- enum `dpdm_mux_cfg_t` {  
`DPDM_MUX_CONN_NONE` = 0x00, `DPDM_MUX_CONN_USB_TOP` = 0x11, `DPDM_MUX_CONN_UART_TOP`  
= 0x22, `DPDM_MUX_CONN_USB_BOT` = 0x44,  
`DPDM_MUX_CONN_UART_BOT` = 0x88, `DPDM_MUX_CONN_USB_TOP_UART` = 0x99, `DPDM_MUX_CONN_USB_BOT_U`  
= 0x66 }
- enum `ccg_supply_t` { `CCG_SUPPLY_VSYS` = 0x00, `CCG_SUPPLY_V5V` = 0x01 }
- enum `ccg_refgen_op_t` {  
`CCG6_VREF_VBUS_UV` = 2, `CCG6_VREF_VBUS_OV` = 3, `CCG6_VREF_CSA_OCP` = 5, `CCG6_VREF_CSA_SCP`  
= 7,  
`CCG6_VREF_RCP_CSA` = 8, `CCG6_VREF_RCP_OVP` = 10 }
- enum `pasc_mode_t` { `PASC_MODE_FF`, `PASC_MODE_PFF`, `PASC_MODE_QR` }

## Functions

- [cpg\\_status\\_t pd\\_hal\\_init](#) (uint8\_t port)
- [bool pd\\_phy\\_deepsleep](#) (uint8\_t port)
- [bool pd\\_phy\\_wakeup](#) (void)
- [void pd\\_hal\\_cleanup](#) (uint8\_t port)
- [void pd\\_hal\\_set\\_supply\\_change\\_evt\\_cb](#) (pd\_supply\_change\_cbk\_t cb)
- [cpg\\_status\\_t pd\\_phy\\_init](#) (uint8\_t port, pd\_phy\_cbk\_t cbk)
- [void pd\\_phy\\_refresh\\_roles](#) (uint8\_t port)
- [void pd\\_phy\\_en\\_unchunked\\_tx](#) (uint8\_t port)
- [void pd\\_phy\\_dis\\_unchunked\\_tx](#) (uint8\_t port)
- [bool pd\\_phy\\_load\\_msg](#) (uint8\_t port, sop\_t sop, uint8\_t retries, uint8\_t dobj\_count, uint32\_t header, bool unchunked, uint32\_t \*buf)
- [bool pd\\_phy\\_send\\_msg](#) (uint8\_t port)
- [void pd\\_phy\\_reset\\_rx\\_tx\\_sm](#) (uint8\_t port)
- [cpg\\_status\\_t pd\\_phy\\_send\\_bist\\_cm2](#) (uint8\_t port)
- [cpg\\_status\\_t pd\\_phy\\_abort\\_bist\\_cm2](#) (uint8\_t port)
- [cpg\\_status\\_t pd\\_phy\\_abort\\_tx\\_msg](#) (uint8\_t port)
- [cpg\\_status\\_t pd\\_phy\\_send\\_reset](#) (uint8\_t port, sop\_t sop)
- [pd\\_packet\\_extd\\_t \\* pd\\_phy\\_get\\_rx\\_packet](#) (uint8\_t port)
- [bool pd\\_phy\\_is\\_busy](#) (uint8\_t port)
- [cpg\\_status\\_t pd\\_typec\\_init](#) (uint8\_t port)
- [cpg\\_status\\_t pd\\_typec\\_start](#) (uint8\_t port)
- [cpg\\_status\\_t pd\\_typec\\_stop](#) (uint8\_t port)
- [void pd\\_typec\\_en\\_rp](#) (uint8\_t port, uint8\_t channel, rp\_term\_t rp\_val)
- [void pd\\_typec\\_dis\\_rp](#) (uint8\_t port, uint8\_t channel)
- [void pd\\_typec\\_en\\_dpslp\\_rp](#) (uint8\_t port)
- [void pd\\_typec\\_dis\\_dpslp\\_rp](#) (uint8\_t port)
- [void pd\\_typec\\_en\\_rd](#) (uint8\_t port, uint8\_t channel)
- [void pd\\_typec\\_rd\\_enable](#) (uint8\_t port)
- [void pd\\_typec\\_dis\\_rd](#) (uint8\_t port, uint8\_t channel)
- [void pd\\_typec\\_en\\_deadbat\\_rd](#) (uint8\_t port)
- [void pd\\_typec\\_snk\\_update\\_trim](#) (uint8\_t port)
- [cc\\_state\\_t pd\\_typec\\_get\\_cc\\_status](#) (uint8\_t port)
- [void pd\\_typec\\_set\\_polarity](#) (uint8\_t port, bool polarity)
- [bool pd\\_is\\_v5v\\_supply\\_on](#) (uint8\_t port)
- [cpg\\_status\\_t pd\\_vconn\\_enable](#) (uint8\_t port, uint8\_t channel)
- [cpg\\_status\\_t pd\\_vconn\\_disable](#) (uint8\_t port, uint8\_t channel)
- [void pd\\_hal\\_vconn\\_ocp\\_enable](#) (uint8\_t port, uint8\_t debounce, PD\_ADC\_CB\_T cb)
- [void pd\\_hal\\_vconn\\_ocp\\_disable](#) (uint8\_t port)
- [bool pd\\_is\\_vconn\\_present](#) (uint8\_t port, uint8\_t channel)
- [uint8\\_t pd\\_adc\\_volt\\_to\\_level](#) (uint8\_t port, PD\_ADC\_ID\_T adc\_id, uint16\_t volt)
- [uint16\\_t pd\\_adc\\_level\\_to\\_volt](#) (uint8\_t port, PD\_ADC\_ID\_T adc\_id, uint8\_t level)
- [uint16\\_t pd\\_adc\\_get\\_vbus\\_voltage](#) (uint8\_t port, PD\_ADC\_ID\_T adc\_id, uint8\_t level)
- [cpg\\_status\\_t pd\\_adc\\_free\\_run\\_ctrl](#) (uint8\_t port, PD\_ADC\_ID\_T adc\_id, PD\_ADC\_INPUT\_T input, uint8\_t level)
- [void pd\\_adc\\_comparator\\_ctrl](#) (uint8\_t port, PD\_ADC\_ID\_T adc\_id, PD\_ADC\_INPUT\_T input, uint8\_t level, PD\_ADC\_INT\_T int\_cfg, PD\_ADC\_CB\_T cb)
- [bool pd\\_adc\\_comparator\\_sample](#) (uint8\_t port, PD\_ADC\_ID\_T adc\_id, PD\_ADC\_INPUT\_T input, uint8\_t level)
- [bool pd\\_adc\\_get\\_comparator\\_status](#) (uint8\_t port, PD\_ADC\_ID\_T adc\_id)
- [uint8\\_t pd\\_adc\\_sample](#) (uint8\_t port, PD\_ADC\_ID\_T adc\_id, PD\_ADC\_INPUT\_T input)
- [uint16\\_t pd\\_adc\\_calibrate](#) (uint8\_t port, PD\_ADC\_ID\_T adc\_id)
- [cpg\\_status\\_t pd\\_adc\\_init](#) (uint8\_t port, PD\_ADC\_ID\_T adc\_id)
- [cpg\\_status\\_t pd\\_adc\\_select\\_vref](#) (uint8\_t port, PD\_ADC\_ID\_T adc\_id, PD\_ADC\_VREF\_T vref\_sel)

- uint8\_t pd\_get\_vbus\_adc\_level (uint8\_t port, PD\_ADC\_ID\_T adc\_id, uint16\_t volt, int8\_t per)
- void pd\_connect\_vbus\_div\_to\_amux (uint8\_t port)
- bool pd\_disconnect\_vbus\_div\_from\_amux (uint8\_t port)
- void pd\_hal\_config\_auto\_toggle (uint8\_t port, bool enable)
- bool pd\_hal\_is\_auto\_toggle\_active (uint8\_t port)
- void pd\_hal\_abort\_auto\_toggle (uint8\_t port)
- void pd\_hal\_typec\_sm\_restart (uint8\_t port)
- void pd\_hal\_set\_reference (uint8\_t port, bool flag)
- uint16\_t pd\_hal\_measure\_vbus\_in (uint8\_t port)
- uint16\_t pd\_hal\_measure\_vbus (uint8\_t port)
- uint16\_t pd\_hal\_measure\_vbus\_cur (uint8\_t port)
- bool pd\_frs\_rx\_enable (uint8\_t port)
- bool pd\_frs\_rx\_disable (uint8\_t port)
- bool pd\_frs\_tx\_enable (uint8\_t port)
- bool pd\_frs\_tx\_disable (uint8\_t port)
- void pd\_hal\_set\_fet\_drive (pd\_fet\_dr\_t pctrl\_drive, pd\_fet\_dr\_t cctrl\_drive)
- void pd\_hal\_dual\_fet\_config (bool dual\_fet, uint8\_t spacing)
- void pd\_reset\_edge\_det (uint8\_t port, bool pgdo\_type)
- void pd\_remove\_internal\_fb\_res\_div (void)
- void pd\_internal\_pfet\_on (uint8\_t port, bool turn\_on\_seq)
- void pd\_internal\_pfet\_off (uint8\_t port, bool turn\_off\_seq)
- void pd\_internal\_cfet\_on (uint8\_t port, bool turn\_on\_seq)
- void pd\_internal\_cfet\_off (uint8\_t port, bool turn\_off\_seq)
- bool pd\_hal\_is\_sink\_fet\_on (uint8\_t port)
- void pd\_internal\_vbus\_discharge\_on (uint8\_t port)
- void pd\_internal\_vbus\_discharge\_off (uint8\_t port)
- void pd\_internal\_vbus\_in\_discharge\_on (uint8\_t port)
- void pd\_internal\_vbus\_in\_discharge\_off (uint8\_t port)
- void pd\_internal\_vbus\_ocp\_en (uint8\_t port, uint8\_t av\_bw, uint8\_t vref\_sel, bool pctrl, uint8\_t mode, uint8\_t debounce\_ms)
- void pd\_internal\_vbus\_ocp\_dis (uint8\_t port, bool pctrl)
- void pd\_internal\_vbus\_load\_change\_isr\_enable (uint8\_t port, uint32\_t cur, uint8\_t filter, vbus\_load\_chg\_cbk\_t cbk)
- void pd\_internal\_vbus\_ovp\_en (uint8\_t port, uint16\_t volt, int8\_t per, PD\_ADC\_CB\_T cb, bool pctrl, vbus\_ovp\_mode\_t mode, uint8\_t filter\_sel)
- void pd\_internal\_vbus\_ovp\_dis (uint8\_t port, bool pctrl)
- void pd\_internal\_vbus\_uvp\_en (uint8\_t port, uint16\_t volt, int8\_t per, PD\_ADC\_CB\_T cb, bool pctrl, vbus\_uvp\_mode\_t mode, uint8\_t filter\_sel)
- void pd\_internal\_vbus\_uvp\_dis (uint8\_t port, bool pctrl)
- void pd\_internal\_vbus\_scp\_en (uint8\_t port, uint32\_t vsense, uint8\_t filter\_sel, bool pctrl, uint8\_t mode)
- void pd\_internal\_vbus\_scp\_dis (uint8\_t port, bool pctrl)
- void pd\_internal\_vbus\_rcp\_en (uint8\_t port, bool pctrl, uint8\_t csa\_det\_en, uint8\_t cmp\_det\_en, uint8\_t ovp←\_det\_en)
- void pd\_internal\_vbus\_rcp\_dis (uint8\_t port, bool pctrl)
- void pd\_internal\_pfet\_soft\_start\_on (uint8\_t port, uint8\_t fet\_status)
- void pd\_internal\_pfet\_soft\_start\_off (uint8\_t port)
- void pd\_internal\_cfet\_soft\_start\_on (uint8\_t port, uint8\_t fet\_status)
- void pd\_internal\_cfet\_soft\_start\_off (uint8\_t port)
- void pd\_fet\_automode\_disable (uint8\_t port, bool pctrl, filter\_id\_t filter\_index)
- void pd\_fet\_automode\_enable (uint8\_t port, bool pctrl, filter\_id\_t filter\_index)
- void pd\_hal\_enable\_internal\_vbus\_mon (bool enable)
- ccg\_status\_t sbu\_switch\_configure (uint8\_t port, sbu\_switch\_state\_t sbu1\_state, sbu\_switch\_state\_t sbu2←\_state)
- sbu\_switch\_state\_t get\_sbu1\_switch\_state (uint8\_t port)
- sbu\_switch\_state\_t get\_sbu2\_switch\_state (uint8\_t port)

- void [aux\\_resistor\\_configure](#) (uint8\_t port, [aux\\_resistor\\_config\\_t](#) aux1\_config, [aux\\_resistor\\_config\\_t](#) aux2\_config)
- [aux\\_resistor\\_config\\_t get\\_aux1\\_resistor\\_config](#) (uint8\_t port)
- [aux\\_resistor\\_config\\_t get\\_aux2\\_resistor\\_config](#) (uint8\_t port)
- void [pd\\_enable\\_vconn\\_comp](#) (void)
- bool [pd\\_get\\_vconn\\_status](#) (void)
- void [pd\\_disconnect\\_ra](#) (void)
- void [ccg\\_set\\_fault\\_cb](#) (PD\_ADC\_CB\_T cb)
- bool [ccg\\_is\\_cdp\\_sm\\_busy](#) (uint8\_t port)
- void [ccg\\_bc\\_dcp\\_en](#) (uint8\_t port)
- bool [ccg\\_bc\\_is\\_cdp](#) (uint8\_t port)
- void [ccg\\_bc\\_cdp\\_en](#) (uint8\_t port)
- void [ccg\\_bc\\_dis](#) (uint8\_t port)
- bool [ccg\\_bc\\_cdp\\_sm](#) (uint8\_t port)
- void [ccg\\_config\\_dp\\_dm\\_mux](#) (uint8\_t port, [dpdm\\_mux\\_cfg\\_t](#) conf)
- void [pd\\_hal\\_set\\_cc\\_ovp\\_pending](#) (uint8\_t port)
- void [pd\\_hal\\_set\\_vbus\\_detach\\_params](#) (PD\_ADC\_ID\_T adc\_id, PD\_ADC\_INPUT\_T adc\_inp)
- PD\_ADC\_ID\_T [pd\\_hal\\_get\\_vbus\\_detach\\_adc](#) (void)
- PD\_ADC\_INPUT\_T [pd\\_hal\\_get\\_vbus\\_detach\\_input](#) (void)
- void [pd\\_hal\\_set\\_vbus\\_mon\\_divider](#) (uint8\_t divider)
- void [pd\\_hal\\_set\\_vbus\\_csa\\_rsense](#) (uint8\_t rsense)
- uint8\_t [pd\\_hal\\_get\\_vbus\\_csa\\_rsense](#) (void)
- void [pd\\_lscsa\\_calc\\_cfg](#) (uint32\_t cur\_10mA, uint8\_t \*gain\_sel, uint8\_t \*vref\_sel)
- [ccg\\_status\\_t pd\\_lscsa\\_cfg](#) ([lscsa\\_app\\_config\\_t](#) lscsa\_app, uint8\_t gain\_sel)
- void [pd\\_cf\\_enable](#) (uint8\_t port, uint32\_t cur)
- void [pd\\_cf\\_disable](#) (uint8\_t port)
- bool [pd\\_cf\\_get\\_status](#) (uint8\_t port)
- void [pd\\_cf\\_mon\\_enable](#) (uint8\_t port, bool reset\_state, [vbus\\_cf\\_cbk\\_t](#) cbk)
- void [pd\\_cf\\_mon\\_disable](#) (uint8\_t port)
- bool [pd\\_set\\_sr\\_comp](#) (uint8\_t port, uint32\_t cur, [filter\\_edge\\_detect\\_cfg\\_t](#) edge, [pd\\_cmp\\_cbk\\_t](#) cbk)
- void [pd\\_stop\\_sr\\_comp](#) (uint8\_t port)
- bool [pd\\_set\\_pfc\\_comp](#) (uint8\_t port, uint32\_t cur, [filter\\_edge\\_detect\\_cfg\\_t](#) edge, [pd\\_cmp\\_cbk\\_t](#) cbk)
- void [pd\\_stop\\_pfc\\_comp](#) (uint8\_t port)
- bool [pd\\_sample\\_pfc\\_comp](#) (uint8\_t port, uint32\_t cur)
- bool [pd\\_cmp\\_get\\_status](#) (uint8\_t port, [filter\\_id\\_t](#) id, bool is\_filtered)
- void [pd\\_srsense\\_enable](#) (uint8\_t port)
- void [pd\\_srsense\\_disable](#) (uint8\_t port)
- void [pd\\_srgdrv\\_enable](#) (uint8\_t port)
- void [pd\\_srgdrv\\_disable](#) (uint8\_t port)
- uint16\_t [pd\\_hal\\_measure\\_line\\_volt](#) (uint8\_t port)
- uint16\_t [pd\\_hal\\_measure\\_ea\\_volt](#) (uint8\_t port)
- void [pd\\_pasc\\_poll\\_task](#) (uint8\_t port)
- void [pd\\_pasc\\_vbus\\_in\\_set\\_vsafe\\_5V](#) (uint8\_t port)
- void [pd\\_pasc\\_vbus\\_in\\_set\\_volt](#) (uint8\_t port, uint16\_t voltmV, PD\_ADC\_CB\_T cbk)
- void [pd\\_pasc\\_vbus\\_in\\_set\\_volt\\_abort](#) (uint8\_t port)
- void [pd\\_pasc\\_start](#) (uint8\_t port)
- uint8\_t [pd\\_pasc\\_get\\_valley](#) (uint8\_t port)
- [ccg\\_status\\_t pd\\_pasc\\_set\\_valley](#) (uint8\_t port, uint8\_t valley, [pasc\\_valley\\_cbk\\_t](#) cb)
- void [pd\\_pasc\\_valley\\_algo\\_enable](#) (uint8\_t port, const [pasc\\_valley\\_table\\_t](#) \*table)
- void [pd\\_pasc\\_ptdrv\\_cont\\_det\\_enable](#) (uint8\_t port, [pasc\\_ptdrv\\_cont\\_cbk\\_t](#) cbk)
- void [pd\\_pasc\\_ptdrv\\_cont\\_det\\_disable](#) (uint8\_t port)
- void [pd\\_pasc\\_lp\\_enable](#) (uint8\_t port)
- void [pd\\_pasc\\_lp\\_disable](#) (uint8\_t port)
- void [pd\\_pasc\\_lp\\_task](#) (uint8\_t port)
- bool [pd\\_pasc\\_lp\\_is\\_active](#) (uint8\_t port)
- bool [pd\\_pasc\\_lp\\_ds\\_allowed](#) (uint8\_t port)
- void [pd\\_pasc\\_send\\_stop\\_signal](#) (uint8\_t port)
- bool [hpdt\\_ctrl1\\_reg\\_check\\_start](#) (uint8\_t port)

## 6.26.1 Detailed Description

CCG PD PHY driver module header file.

## 6.26.2 Macro Definition Documentation

### 6.26.2.1 AUTO\_CTRL\_MESSAGE\_GOODCRC\_MASK\_CFG

```
#define AUTO_CTRL_MESSAGE_GOODCRC_MASK_CFG (0xFFFFFFFFDu)
```

Select types of PD control messages for which GoodCRC response should be sent. All except GoodCRC are enabled.

### 6.26.2.2 AUTO\_DATA\_MESSAGE\_GOODCRC\_MASK\_CFG

```
#define AUTO_DATA_MESSAGE_GOODCRC_MASK_CFG (0xFFFFFFFFFu)
```

Select types of PD data messages for which GoodCRC response should be sent.

### 6.26.2.3 AUTO\_EXTD\_MESSAGE\_GOODCRC\_MASK\_CFG

```
#define AUTO_EXTD_MESSAGE_GOODCRC_MASK_CFG (0xFFFFFFFFFu)
```

Select types of PD extended data messages for which GoodCRC response should be sent.

### 6.26.2.4 EXPECTED\_GOOD\_CRC\_CLEAR\_MASK

```
#define EXPECTED_GOOD_CRC_CLEAR_MASK (0xF1FFu)
```

Mask to clear expected GoodCRC header field.

### 6.26.2.5 EXPECTED\_GOOD\_CRC\_HDR\_DIFF\_MASK\_REV3

```
#define EXPECTED_GOOD_CRC_HDR_DIFF_MASK_REV3 (PD_MSG_HDR_REV2_IGNORE_MASK)
```

Mask representing additional header bits that should be matched for expected GoodCRC detection during PD 3.0 operation.

### 6.26.2.6 EXPECTED\_GOOD\_CRC\_HDR\_MASK

```
#define EXPECTED_GOOD_CRC_HDR_MASK (0x7E0Fu)
```

Mask representing header bits that should be matched for expected GoodCRC detection.

### 6.26.2.7 PD\_MSG\_HDR\_REV2\_IGNORE\_MASK

```
#define PD_MSG_HDR_REV2_IGNORE_MASK (0x8010)
```

Mask representing bits that should be ignored in a PD 2.0 message header.

### 6.26.2.8 PDSS\_CC\_FILT\_CYCLES

```
#define PDSS_CC_FILT_CYCLES (10)
```

Number of LF clock cycles used for filtering CC comparator output.

### 6.26.2.9 PDSS\_DRP\_HIGH\_PERIOD\_MS

```
#define PDSS_DRP_HIGH_PERIOD_MS (30u)
```

Rp assert period in ms for hardware DRP toggle.

### 6.26.2.10 PDSS\_DRP\_HIGH\_PERIOD\_VAL

```
#define PDSS_DRP_HIGH_PERIOD_VAL (1100u)
```

Rp assert period in LF clock cycles.

### 6.26.2.11 PDSS\_DRP\_TOGGLE\_PERIOD\_MS

```
#define PDSS_DRP_TOGGLE_PERIOD_MS (75u)
```

Hardware based DRP toggle period in ms.

### 6.26.2.12 PDSS\_DRP\_TOGGLE\_PERIOD\_VAL

```
#define PDSS_DRP_TOGGLE_PERIOD_VAL (2500u)
```

Hardware DRP toggle period in LF clock cycles.

### 6.26.2.13 PDSS\_MAX\_RX\_MEM\_HALF\_SIZE

```
#define PDSS_MAX_RX_MEM_HALF_SIZE (8u)
```

Half of PD receive FIFO size in 4-byte words.

### 6.26.2.14 PDSS\_MAX\_RX\_MEM\_SIZE

```
#define PDSS_MAX_RX_MEM_SIZE (16u)
```

PD receive FIFO size in 4-byte words.

### 6.26.2.15 PDSS\_MAX\_TX\_MEM\_HALF\_SIZE

```
#define PDSS_MAX_TX_MEM_HALF_SIZE (8u)
```

Half of PD transmit FIFO size in 4-byte words.

### 6.26.2.16 PDSS\_MAX\_TX\_MEM\_SIZE

```
#define PDSS_MAX_TX_MEM_SIZE (16u)
```

PD transmit FIFO size in 4-byte words.

#### 6.26.2.17 PGDO\_PD\_ISINK\_TERMINAL\_VAL

```
#define PGDO_PD_ISINK_TERMINAL_VAL (0x3F)
```

Maximum value of PGDO pull-down current sink.

#### 6.26.2.18 PGDO\_PD\_ISINK\_TERMINAL\_VAL\_1

```
#define PGDO_PD_ISINK_TERMINAL_VAL_1 (0x1F)
```

Maximum value of PGDO pull-down current sink extension.

#### 6.26.2.19 RX\_CNT\_MAX\_VAL

```
#define RX_CNT_MAX_VAL (0x0Fu)
```

1/16 of the number of RX clock cycles required by CC receiver to detect idle state.

#### 6.26.2.20 RX\_UI\_BOUNDARY\_DELTA\_VAL

```
#define RX_UI_BOUNDARY_DELTA_VAL (0x02u)
```

CC receiver voltage threshold selection. Must be set to 2 for a 12 MHz RX CLK.

#### 6.26.2.21 VBUS\_OCP\_GPIO\_ACTIVE\_HIGH

```
#define VBUS_OCP_GPIO_ACTIVE_HIGH (1u)
```

GPIO high indicates fault condition.

#### 6.26.2.22 VBUS\_OCP\_GPIO\_ACTIVE\_LOW

```
#define VBUS_OCP_GPIO_ACTIVE_LOW (0u)
```

GPIO low indicates fault condition.

#### 6.26.2.23 VBUS\_OCP\_MODE\_EXT

```
#define VBUS_OCP_MODE_EXT (0u)
```

OCP through external hardware.

#### 6.26.2.24 VBUS\_OCP\_MODE\_INT

```
#define VBUS_OCP_MODE_INT (1u)
```

Internal OCP without software debounce or hardware gate control.

#### 6.26.2.25 VBUS\_OCP\_MODE\_INT\_AUTOCTRL

```
#define VBUS_OCP_MODE_INT_AUTOCTRL (2u)
```

Internal OCP with hardware gate control.



## 6.26.2.26 VBUS\_OCP\_MODE\_INT\_SW\_DB

```
#define VBUS_OCP_MODE_INT_SW_DB (3u)
```

Internal OCP with software debounce.

## 6.26.2.27 VBUS\_OCP\_MODE\_POLLING

```
#define VBUS_OCP_MODE_POLLING (4u)
```

Solution level polling based OCP.

## 6.26.3 Typedef Documentation

## 6.26.3.1 pasc\_ptdrv\_cont\_cbk\_t

```
pasc_ptdrv_cont_cbk_t
```

Callback function used to provide notification on PTDRV contention.

## Parameters

|             |                                     |
|-------------|-------------------------------------|
| <i>port</i> | PD port associated with the supply. |
|-------------|-------------------------------------|

**Applicable devices:** PAG1S

## 6.26.3.2 pasc\_valley\_cbk\_t

```
pasc_valley_cbk_t
```

Callback function used to provide notification on valley change completion.

## Parameters

|               |                                                                    |
|---------------|--------------------------------------------------------------------|
| <i>port</i>   | PD port associated with the supply.                                |
| <i>status</i> | Status of the transition whether it completed successfully or not. |

## 6.26.3.3 PD\_ADC\_CB\_T

```
PD_ADC_CB_T
```

PD ADC comparator interrupt callback type. This callback is called when the desired ADC event/interrupt occurs. Available events:

- true : Input voltage is higher than the reference.
- false : Input voltage is lower than the reference.

## Parameters

|                 |                                          |
|-----------------|------------------------------------------|
| <i>port</i>     | PD port on which the ADC event occurred. |
| <i>comp_out</i> | Specifies the type of event.             |

#### 6.26.3.4 pd\_cmp\_cbk\_t

pd\_cmp\_cbk\_t

Comparator interrupt callback function.

##### Parameters

|              |                                      |
|--------------|--------------------------------------|
| <i>port</i>  | PD port on which the event occurred. |
| <i>state</i> | State of the comparator.             |

#### 6.26.3.5 pd\_phy\_cbk\_t

pd\_phy\_cbk\_t

PD PHY callback prototype. This function will be used to notify the stack about PHY events.

##### Parameters

|              |                                          |
|--------------|------------------------------------------|
| <i>port</i>  | PD port on which the PHY event occurred. |
| <i>event</i> | Type of PD PHY event.                    |

#### 6.26.3.6 pd\_supply\_change\_cbk\_t

pd\_supply\_change\_cbk\_t

Callback function used to provide notification about input supply changes.

##### Parameters

|                        |                                                                        |
|------------------------|------------------------------------------------------------------------|
| <i>port</i>            | PD port associated with the supply.                                    |
| <i>supply↔<br/>_id</i> | ID of the supply on which change is detected.                          |
| <i>present</i>         | Whether the identified supply is now present (true) or absent (false). |

#### 6.26.3.7 vbus\_cf\_cbk\_t

vbus\_cf\_cbk\_t

VBus current foldback callback function.

##### Parameters

|                 |                                      |
|-----------------|--------------------------------------|
| <i>port</i>     | PD port on which the event occurred. |
| <i>cf_state</i> | Whether in CF mode or not.           |

### 6.26.3.8 vbus\_load\_chg\_cbk\_t

vbus\_load\_chg\_cbk\_t

Callback function used to provide notification on VBUS load change.

#### Parameters

|             |                                     |
|-------------|-------------------------------------|
| <i>port</i> | PD port associated with the supply. |
|-------------|-------------------------------------|

**Applicable devices:** PAG1S

## 6.26.4 Enumeration Type Documentation

### 6.26.4.1 aux\_resistor\_config\_t

enum `aux_resistor_config_t`

Enum to hold resistor configuration for AUX1 and AUX2. Values assigned are the bit position of corresponding configuration in `sbu_ctrl` register. Only applicable to CCG3 and CCG5 devices.

#### Enumerator

|                          |                                 |
|--------------------------|---------------------------------|
| AUX_NO_RESISTOR          | No resistor.                    |
| AUX_1_1MEG_PU_RESISTOR   | AUX1 1M0hm Pullup resistor.     |
| AUX_1_100K_PD_RESISTOR   | AUX1 100KOhm Pulldown resistor. |
| AUX_1_470K_PD_RESISTOR   | AUX1 470KOhm Pulldown resistor. |
| AUX_2_100K_PU_RESISTOR   | AUX2 100KOhm Pullup resistor.   |
| AUX_2_4P7MEG_PD_RESISTOR | AUX2 4.7M0hm Pulldown resistor. |
| AUX_2_1MEG_PD_RESISTOR   | AUX2 1M0hm Pulldown resistor.   |
| AUX_MAX_RESISTOR_CONFIG  | Not supported.                  |

### 6.26.4.2 ccg\_refgen\_op\_t

enum `ccg_refgen_op_t`

List of refgen outputs on the CCG6 device.

#### Enumerator

|                   |                                                       |
|-------------------|-------------------------------------------------------|
| CCG6_VREF_VBUS_UV | Vref used for UVP on CCG6.                            |
| CCG6_VREF_VBUS_OV | Vref used for OVP on CCG6.                            |
| CCG6_VREF_CSA_OCP | Vref used for CSA OCP on CCG6.                        |
| CCG6_VREF_CSA_SCP | Vref used for CSA SCP on CCG6.                        |
| CCG6_VREF_RCP_CSA | Vref used for Reverse current sensing on CCG6.        |
| CCG6_VREF_RCP_OVP | Vref used for RCP based on VBus over-voltage on CCG6. |

### 6.26.4.3 ccg\_supply\_t

enum `ccg_supply_t`

List of power supplies input to and monitored by the CCG5, CCG5C and CCG6 devices. This does not include the VBus supply which is monitored by all CCG devices as required by the Type-C and USB-PD specifications.

#### Enumerator

|                 |                                                                       |
|-----------------|-----------------------------------------------------------------------|
| CCG_SUPPLY_VSYS | Vsys supply input which powers the device.                            |
| CCG_SUPPLY_V5V  | V5V input used to derive the VConn supply from. Can be port-specific. |

### 6.26.4.4 comp\_id\_t

enum `comp_id_t`

List of dedicated comparators supported by the PD block on various CCG devices. The actual comparators supported vary across device families. See comments associated with each comparator ID for list of devices that support this comparator.

#### Enumerator

|                        |                                                         |
|------------------------|---------------------------------------------------------|
| COMP_ID_UV             | UV comparator. Common for CCG3PA, CCG5, CCG5C and CCG6. |
| COMP_ID_OV             | OV comparator. Common for CCG3PA, CCG5, CCG5C and CCG6. |
| COMP_ID_VBUS_C_MON     | VBUS_C_MON comparator. CCG3PA and CCG3PA2 only.         |
| COMP_ID_VBUS_MON       | VBUS_MON comparator. CCG5, CCG5C and CCG6 only.         |
| COMP_ID_VBUS_DISCHARGE | Discharge comparator. CCG3PA and CCG3PA2 only.          |
| COMP_ID_VSYS_DET       | VSYS detection. CCG5, CCG5C and CCG6 only.              |
| COMP_ID_LSCSA_SCP      | SCP comparator. CCG3PA and CCG3PA2 only.                |
| COMP_ID_DP_DETACH      | D+ voltage comparator. CCG6 only.                       |
| COMP_ID_DM_DETACH      | D- voltage comparator. CCG6 only.                       |
| COMP_ID_LSCSA_OCP      | OCP comparator. PAG1S only.                             |
| COMP_ID_LSCSA_PFC      | PFC comparator. CCG3PA and CCG3PA2 only.                |
| COMP_ID_VSRC_NEW_P     | VSRC_NEW comparator. CCG3PA and CCG3PA2 only.           |
| COMP_ID_VSRC_NEW_M     | VSRC_NEW comparator. CCG3PA and CCG3PA2 only.           |
| COMP_ID_MAX            | End of comparator list.                                 |

### 6.26.4.5 comp\_tr\_id\_t

enum `comp_tr_id_t`

IDs of trim enabled comparators in CCG3PA and CCG3PA2 devices.

#### Enumerator

|                |                                         |
|----------------|-----------------------------------------|
| COMP_TR_ID_SR  | SR comparator. CCG3PA and CCG3PA2 only. |
| COMP_TR_ID_MAX | End of comparator list.                 |

## 6.26.4.6 dpdm\_mux\_cfg\_t

enum `dpdm_mux_cfg_t`

List of possible settings for the DP/DM MUX on the CCG5, CCG5C and CCG6 devices.

## Enumerator

|                            |                                                                  |
|----------------------------|------------------------------------------------------------------|
| DPDM_MUX_CONN_NONE         | No connections enabled through the DPDM Mux.                     |
| DPDM_MUX_CONN_USB_TOP      | Connect D+/D- to D+/D- on the top side of the connector.         |
| DPDM_MUX_CONN_UART_TOP     | Connect UART TX/RX to D+/D- on the top side of the connector.    |
| DPDM_MUX_CONN_USB_BOT      | Connect D+/D- to D+/D- on the bottom side of the connector.      |
| DPDM_MUX_CONN_UART_BOT     | Connect UART TX/RX to D+/D- on the bottom side of the connector. |
| DPDM_MUX_CONN_USB_TOP_UART | Connect D+/D- to top and UART TX/RX to bottom side.              |
| DPDM_MUX_CONN_USB_BOT_UART | Connect D+/D- to bottom and UART TX/RX to top side.              |

## 6.26.4.7 filter\_edge\_detect\_cfg\_t

enum `filter_edge_detect_cfg_t`

List of edge triggered interrupt based on the filter output.

## Enumerator

|                            |                                    |
|----------------------------|------------------------------------|
| FILTER_CFG_POS_DIS_NEG_DIS | Interrupt disabled.                |
| FILTER_CFG_POS_DIS_NEG_EN  | Negative edge detection interrupt. |
| FILTER_CFG_POS_EN_NEG_DIS  | Positive edge detection interrupt. |
| FILTER_CFG_POS_EN_NEG_EN   | Both edge detection interrupt.     |
| FILTER_CFG_MAX             | Invalid interrupt configuration.   |

## 6.26.4.8 filter\_id\_t

enum `filter_id_t`

The outputs of the comparators listed in `comp_id_t` and `comp_tr_id_t` can be passed through a filter for debouncing. As with the comparators, the filters supported vary across device families. This enumeration lists various filters supported and the comments indicate the devices that support them.

## Enumerator

|                     |                                                                                                          |
|---------------------|----------------------------------------------------------------------------------------------------------|
| FILTER_ID_UV        | UV comparator filter, common for CCG3PA, CCG3PA2, CCG5, CCG5C and CCG6. Can run based on HF or LF clock. |
| FILTER_ID_OV        | OV comparator filter, common for CCG3PA, CCG3PA2, CCG5, CCG5C and CCG6. Can run based on HF or LF clock. |
| FILTER_ID_DISCH_EN  | Discharge enable filter for CCG3PA and CCG3PA2. Can run based on HF or LF clock.                         |
| FILTER_ID_LSCSA_OCP | OCP filter for PAG1S. Can run based on HF or LF clock.                                                   |
| FILTER_ID_LSCSA_PFC | PFC filter for PAG1S. Can run based on HF or LF clock.                                                   |

## Enumerator

|                      |                                                               |
|----------------------|---------------------------------------------------------------|
| FILTER_ID_LSCSA_SR   | SR filter for PAG1S. Can run based on HF or LF clock.         |
| FILTER_ID_VSRC_NEW_P | VSRC_NEW_P filter for PAG1S. Can run based on HF or LF clock. |
| FILTER_ID_VSRC_NEW_M | VSRC_NEW_M filter for PAG1S. Can run based on HF or LF clock. |
| FILTER_ID_PDS_SCP    | Virtual number for SCP as it is to be configured differently. |
| FILTER_ID_MAX        | Number of supported filters.                                  |

## 6.26.4.9 frs\_tx\_source\_t

```
enum frs_tx_source_t
```

Enum to hold various tx sources of FR-SWAP.

## Enumerator

|                    |                                                        |
|--------------------|--------------------------------------------------------|
| FRS_TX_SOURCE_CPU  | FRS signal sent through FW intervention.               |
| FRS_TX_SOURCE_GPIO | FRS signal triggered based on GPIO input.              |
| FRS_TX_SOURCE_ADC1 | FRS signal triggered based on ADC-0 comparator output. |
| FRS_TX_SOURCE_ADC2 | FRS signal triggered based on ADC-1 comparator output. |

## 6.26.4.10 lscsa\_app\_config\_t

```
enum lscsa_app_config_t
```

Enumeration of all possible Low Side CSA applications. Only applicable to CCG3PA and CCG3PA2.

## Enumerator

|                        |                                              |
|------------------------|----------------------------------------------|
| LSCSA_OCP_CONFIG       | OCP comparator gain control CCG3PA only.     |
| LSCSA_EA_CONFIG        | EA comparator gain control CCG3PA only.      |
| LSCSA_PFC_OFF_CONFIG   | PFC OFF comparator gain control CCG3PA only. |
| LSCSA_PFC_ON_CONFIG    | PFC ON comparator gain control CCG3PA only.  |
| LSCSA_SR_OFF_CONFIG    | SR OFF comparator gain control CCG3PA only.  |
| LSCSA_SR_ON_CONFIG     | SR ON comparator gain control CCG3PA only.   |
| LSCSA_MAX_CONFIG_VALUE | End of comparator list.                      |

## 6.26.4.11 pasc\_mode\_t

```
enum pasc_mode_t
```

List of secondary control PWM operation modes.

## Enumerator

|               |                                                 |
|---------------|-------------------------------------------------|
| PASC_MODE_FF  | PWM operation in fixed frequency mode.          |
| PASC_MODE_PFF | PWM operation in pseudo fixed frequency mode.   |
| PASC_MODE_QR  | PWM operation in Quasi Resonant frequency mode. |

## 6.26.4.12 PD\_ADC\_ID\_T

enum [PD\\_ADC\\_ID\\_T](#)

ADC block IDs on the CCG device.

## Enumerator

|                |                                                               |
|----------------|---------------------------------------------------------------|
| PD_ADC_ID_0    | ADC-0 in the PD block. Supported by all devices.              |
| PD_ADC_ID_1    | ADC-1 in the PD block. Not supported on CCG5, CCG5C and CCG6. |
| PD_ADC_NUM_ADC | Maximum number of ADCs in the PD block.                       |

## 6.26.4.13 PD\_ADC\_INPUT\_T

enum [PD\\_ADC\\_INPUT\\_T](#)

Enumeration of PD ADC input sources. Refer to CCG device datasheet and TRM for more details.

## Enumerator

|                      |                                 |
|----------------------|---------------------------------|
| PD_ADC_INPUT_AMUX_A  | AMUX_A bus.                     |
| PD_ADC_INPUT_AMUX_B  | AMUX_B bus.                     |
| PD_ADC_INPUT_BANDGAP | BANDGAP input.                  |
| PD_ADC_INPUT_BJT     | BJT.                            |
| PD_ADC_NUM_INPUT     | Number of ADC inputs available. |

## 6.26.4.14 PD\_ADC\_INT\_T

enum [PD\\_ADC\\_INT\\_T](#)

PD comparator interrupt configuration enumeration. Note: These are the settings for INTR\_1\_CFG ADC output, not ADC\_SAR\_CTRL.

## Enumerator

|                     |                                       |
|---------------------|---------------------------------------|
| PD_ADC_INT_DISABLED | Comparator interrupt disabled.        |
| PD_ADC_INT_FALLING  | Comparator interrupt on falling edge. |
| PD_ADC_INT_RISING   | Comparator interrupt on rising edge.  |
| PD_ADC_INT_BOTH     | Comparator interrupt on either edge.  |

## 6.26.4.15 PD\_ADC\_VREF\_T

enum [PD\\_ADC\\_VREF\\_T](#)

ADC block reference voltage selection for the CCG device.

## Enumerator

|                  |                                                       |
|------------------|-------------------------------------------------------|
| PD_ADC_VREF_PROG | Programmable reference voltage from the RefGen block. |
| PD_ADC_VREF_VDDD | VDDD supply used as ADC reference voltage.            |

## 6.26.4.16 pd\_fet\_dr\_t

enum [pd\\_fet\\_dr\\_t](#)

Enum to hold power FET gate drive modes.

## Enumerator

|                       |                                             |
|-----------------------|---------------------------------------------|
| PD_FET_DR_ACTIVE_HIGH | Drive GPIO high to enable FET.              |
| PD_FET_DR_ACTIVE_LOW  | Drive GPIO low to enable FET.               |
| PD_FET_DR_N_JN_FET    | Use dedicated N-channel MOSFET gate driver. |
| PD_FET_DR_P_JN_FET    | Use dedicated P-channel MOSFET gate driver. |

## 6.26.4.17 pd\_phy\_evt\_t

enum [pd\\_phy\\_evt\\_t](#)

PD PHY state enumeration.

## Enumerator

|                             |                                                         |
|-----------------------------|---------------------------------------------------------|
| PD_PHY_EVT_TX_MSG_COLLISION | Bus busy at message transmission.                       |
| PD_PHY_EVT_TX_MSG_PHY_IDLE  | Bus idle, ready for message transmission.               |
| PD_PHY_EVT_TX_MSG_FAILED    | Message transmission was not successful.                |
| PD_PHY_EVT_TX_MSG_SUCCESS   | Message transmission was successful.                    |
| PD_PHY_EVT_TX_RST_COLLISION | Bus busy just before reset transmission.                |
| PD_PHY_EVT_TX_RST_SUCCESS   | Reset transmission was successful.                      |
| PD_PHY_EVT_RX_MSG           | Message received.                                       |
| PD_PHY_EVT_RX_MSG_CMPLT     | Message received and GoodCRC sent aka collision type 3. |
| PD_PHY_EVT_RX_RST           | Reset was received.                                     |
| PD_PHY_EVT_FRS_SIG_RCVD     | FRS signal was received.                                |
| PD_PHY_EVT_FRS_SIG_SENT     | FRS signal was transmitted.                             |
| PD_PHY_EVT_CRC_ERROR        | Message recieved with CRC error.                        |



## 6.26.4.18 sbu\_switch\_state\_t

enum [sbu\\_switch\\_state\\_t](#)

Enum to hold SBU connection state for CCG3/CCG5. CCG3 and CCG5 provide internal switch to route SBU1/SBU2 signals to AUX\_P/AUX\_N, LSTX/LSRX, or Isolate.

## Enumerator

|                   |                               |
|-------------------|-------------------------------|
| SBU_NOT_CONNECTED | SBU pin is isolated.          |
| SBU_CONNECT_AUX1  | Connect SBU pin to AUX_P.     |
| SBU_CONNECT_AUX2  | Connect SBU pin to AUX_N.     |
| SBU_CONNECT_LSTX  | Connect SBU pin to LSTX.      |
| SBU_CONNECT_LSRX  | Connect SBU pin to LSRX.      |
| SBU_MAX_STATE     | Invalid value: not supported. |

## 6.26.4.19 vbus\_ovp\_mode\_t

enum [vbus\\_ovp\\_mode\\_t](#)

CCG OVP modes enumeration.

## Enumerator

|                             |                                                                                                                 |
|-----------------------------|-----------------------------------------------------------------------------------------------------------------|
| VBUS_OVP_MODE_ADC           | OVP using CCG internal ADC.                                                                                     |
| VBUS_OVP_MODE_UVOV          | OVP using the UVOV block on CCGx. Actual gate control is done by firmware. UVOV block is not supported on CCG4. |
| VBUS_OVP_MODE_UVOV_AUTOCTRL | OVP using the OVOV block with automatic gate driver control. UVOV block is not supported on CCG4.               |

## 6.26.4.20 vbus\_uvp\_mode\_t

enum [vbus\\_uvp\\_mode\\_t](#)

CCG UVP modes enumeration.

## Enumerator

|                                 |                                                                                                |
|---------------------------------|------------------------------------------------------------------------------------------------|
| VBUS_UVP_MODE_ADC               | UVP using CCG internal ADC.                                                                    |
| VBUS_UVP_MODE_INT_COMP          | UVP using internal comparator. Actual gate control is done by firmware. Not supported on CCG4. |
| VBUS_UVP_MODE_INT_COMP_AUTOCTRL | UVP using internal comparator with automatic gate driver control. Not supported on CCG4.       |

## 6.26.5 Function Documentation

### 6.26.5.1 aux\_resistor\_configure()

```
void aux_resistor_configure (
 uint8_t port,
 aux_resistor_config_t aux1_config,
 aux_resistor_config_t aux2_config)
```

Configure AUX1 and AUX2 resistor configuration.

#### Parameters

|                    |                                     |
|--------------------|-------------------------------------|
| <i>port</i>        | PD port to be configured.           |
| <i>aux1_config</i> | Desired AUX1 resistor configuration |
| <i>aux2_config</i> | Desired AUX2 resistor configuration |

#### Returns

None

**Applicable devices:** CCG3.

### 6.26.5.2 ccg\_bc\_cdp\_en()

```
void ccg_bc_cdp_en (
 uint8_t port)
```

Enable CDP operation on the PD port. The CDP state machine should be called periodically until the HAL reports state machine idle.

#### Parameters

|             |                                                      |
|-------------|------------------------------------------------------|
| <i>port</i> | PD port on which CDP state machine is to be started. |
|-------------|------------------------------------------------------|

#### Returns

None

**Applicable devices:** CCG5.

### 6.26.5.3 ccg\_bc\_cdp\_sm()

```
bool ccg_bc_cdp_sm (
 uint8_t port)
```

CDP state machine function.

#### Parameters

|             |                            |
|-------------|----------------------------|
| <i>port</i> | PD port for CDP operation. |
|-------------|----------------------------|

#### Returns

Returns whether CDP state machine is still active.

**Applicable devices:** CCG5.

#### 6.26.5.4 ccg\_bc\_dcp\_en()

```
void ccg_bc_dcp_en (
 uint8_t port)
```

Enable DCP operation on the PD port.

##### Parameters

|             |                        |
|-------------|------------------------|
| <i>port</i> | PD port to be updated. |
|-------------|------------------------|

##### Returns

None

**Applicable devices:** CCG5.

#### 6.26.5.5 ccg\_bc\_dis()

```
void ccg_bc_dis (
 uint8_t port)
```

Disable all BC 1.2 operation on the PD port.

##### Parameters

|             |                     |
|-------------|---------------------|
| <i>port</i> | Port to be updated. |
|-------------|---------------------|

##### Returns

None

**Applicable devices:** CCG5.

#### 6.26.5.6 ccg\_bc\_is\_cdp()

```
bool ccg_bc_is_cdp (
 uint8_t port)
```

Check whether the BC 1.2 port is configured as CDP. This only checks for configuration and not actual operation as CDP.

##### Parameters

|             |                        |
|-------------|------------------------|
| <i>port</i> | PD port to be queried. |
|-------------|------------------------|

##### Returns

true if the port is configured as CDP, false otherwise.

**Applicable devices:** CCG5.

#### 6.26.5.7 ccg\_config\_dp\_dm\_mux()

```
void ccg_config_dp_dm_mux (
```

```
uint8_t port,
dpdm_mux_cfg_t conf)
```

Configure the DP/DM MUX on the CCG device as required.

#### Parameters

|             |                                |
|-------------|--------------------------------|
| <i>port</i> | PD port to be configured.      |
| <i>conf</i> | Type of connection to be made. |

#### Returns

None

**Applicable devices:** CCG5, CCG5C, CCG6.

#### 6.26.5.8 ccg\_is\_cdp\_sm\_busy()

```
bool ccg_is_cdp_sm_busy (
uint8_t port)
```

Check whether CDP state machine is active.

#### Parameters

|             |                        |
|-------------|------------------------|
| <i>port</i> | PD port to be checked. |
|-------------|------------------------|

#### Returns

True if CDP state machine is active, false otherwise.

**Applicable devices:** CCG5.

#### 6.26.5.9 ccg\_set\_fault\_cb()

```
void ccg_set_fault_cb (
PD_ADC_CB_T cb)
```

Register a callback for notification of CC/SBU fault conditions.

#### Parameters

|           |                            |
|-----------|----------------------------|
| <i>cb</i> | Callback function pointer. |
|-----------|----------------------------|

#### Returns

None

**Applicable devices:** CCG5, CCG5C, CCG6.

#### 6.26.5.10 get\_aux1\_resistor\_config()

```
aux_resistor_config_t get_aux1_resistor_config (
uint8_t port)
```

Returns AUX1 resistor configuration.

## Parameters

|             |                        |
|-------------|------------------------|
| <i>port</i> | PD port to be queried. |
|-------------|------------------------|

## Returns

AUX1 resistor config

**Applicable devices:** CCG3.

6.26.5.11 `get_aux2_resistor_config()`

```
aux_resistor_config_t get_aux2_resistor_config (
 uint8_t port)
```

Returns AUX2 resistor configuration.

## Parameters

|             |                        |
|-------------|------------------------|
| <i>port</i> | PD port to be queried. |
|-------------|------------------------|

## Returns

AUX2 resistor config

**Applicable devices:** CCG3.

6.26.5.12 `get_sbu1_switch_state()`

```
sbu_switch_state_t get_sbu1_switch_state (
 uint8_t port)
```

Returns SBU1 switch state.

## Parameters

|             |                        |
|-------------|------------------------|
| <i>port</i> | PD port to be queried. |
|-------------|------------------------|

## Returns

SBU1 switch state

**Applicable devices:** CCG3, CCG5, CCG5C, CCG6.

6.26.5.13 `get_sbu2_switch_state()`

```
sbu_switch_state_t get_sbu2_switch_state (
 uint8_t port)
```

Returns SBU2 switch state.

## Parameters

|             |                        |
|-------------|------------------------|
| <i>port</i> | PD port to be queried. |
|-------------|------------------------|

**Returns**

SBU2 switch state

**Applicable devices:** CCG3, CCG5, CCG5C, CCG6.

**6.26.5.14 hpd\_t\_ctrl1\_reg\_check\_start()**

```
bool hpd_t_ctrl1_reg_check_start (
 uint8_t port)
```

Returns HPD state machine status.

**Parameters**

|             |              |
|-------------|--------------|
| <i>port</i> | USB-PD port. |
|-------------|--------------|

**Returns**

HPD state machine status

**6.26.5.15 pd\_adc\_calibrate()**

```
uint16_t pd_adc_calibrate (
 uint8_t port,
 PD_ADC_ID_T adc_id)
```

This function calibrates the specified ADC for operation.

This function calibrates the specified ADC by identifying the VDDD voltage for reference. It should be noted that by calling the function, the previously calculated threshold levels may have to be changed based on the VDDD reading. The VDDD level is calculated based on the bandgap voltage which is expected to be constant.

**Parameters**

|               |                                                                |
|---------------|----------------------------------------------------------------|
| <i>port</i>   | Port index. Caller should ensure to provide only valid values. |
| <i>adc_id</i> | ADC ID.                                                        |

**Returns**

Returns the VDDD value in mV after calibration.

**Applicable devices:** CCG3, CCG4.

**6.26.5.16 pd\_adc\_comparator\_ctrl()**

```
void pd_adc_comparator_ctrl (
 uint8_t port,
 PD_ADC_ID_T adc_id,
 PD_ADC_INPUT_T input,
 uint8_t level,
 PD_ADC_INT_T int_cfg,
 PD_ADC_CB_T cb)
```

This function configures the ADC for comparator functionality with the requested threshold, and registers a callback which shall be called when the comparator output changes.

This function configures the ADC block as a comparator. The function takes the input to be configured and the ADC comparator threshold. It also takes a callback. If the callback is not NULL, then the threshold is configured and interrupts are enabled. If the callback is NULL, then the ADC / comparator is set to the low power state and interrupts are disabled.

#### Parameters

|                     |                                                                |
|---------------------|----------------------------------------------------------------|
| <i>port</i>         | Port index. Caller should ensure to provide only valid values. |
| <i>adc↔<br/>_id</i> | ADC ID.                                                        |
| <i>input</i>        | ADC input source.                                              |
| <i>level</i>        | Comparator level.                                              |
| <i>int_cfg</i>      | Interrupt configuration.                                       |
| <i>cb</i>           | Callback to be called on interrupt.                            |

#### Returns

None

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

#### 6.26.5.17 pd\_adc\_comparator\_sample()

```
bool pd_adc_comparator_sample (
 uint8_t port,
 PD_ADC_ID_T adc_id,
 PD_ADC_INPUT_T input,
 uint8_t level)
```

This function temporarily configures the comparator as requested and checks whether the input exceeds the specified digital level.

This function restores the comparator to its previous state after operation. This is useful when the comparator is already configured to function with a certain input and level with interrupt and another reading needs to be done without having to re-configure the block after the sampling.

#### Parameters

|                     |                                                                |
|---------------------|----------------------------------------------------------------|
| <i>port</i>         | Port index. Caller should ensure to provide only valid values. |
| <i>adc↔<br/>_id</i> | ADC ID.                                                        |
| <i>input</i>        | ADC input source.                                              |
| <i>level</i>        | Value to compare the input voltage against.                    |

#### Returns

Returns true if voltage > level, false otherwise.

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

#### 6.26.5.18 pd\_adc\_free\_run\_ctrl()

```
ccg_status_t pd_adc_free_run_ctrl (
```



```
uint8_t port,
PD_ADC_ID_T adc_id,
PD_ADC_INPUT_T input,
uint8_t level)
```

This function configures the ADC for comparator functionality with the requested threshold with no interrupts enabled.

#### Parameters

|                     |                                                                |
|---------------------|----------------------------------------------------------------|
| <i>port</i>         | Port index. Caller should ensure to provide only valid values. |
| <i>adc↔<br/>_id</i> | ADC ID.                                                        |
| <i>input</i>        | ADC input source.                                              |
| <i>level</i>        | Comparator level.                                              |

#### Returns

ccg\_status\_t

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

#### 6.26.5.19 pd\_adc\_get\_comparator\_status()

```
bool pd_adc_get_comparator_status (
 uint8_t port,
 PD_ADC_ID_T adc_id)
```

This function gets the current comparator status.

This function does not configure the ADC / comparator. It just returns the current state of the comparator. If true is returned, then the input voltage is greater than the reference and if false, the input voltage is lower than the reference.

#### Parameters

|                     |                                                                |
|---------------------|----------------------------------------------------------------|
| <i>port</i>         | Port index. Caller should ensure to provide only valid values. |
| <i>adc↔<br/>_id</i> | ADC ID.                                                        |

#### Returns

Returns the comparator output.

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

#### 6.26.5.20 pd\_adc\_get\_vbus\_voltage()

```
uint16_t pd_adc_get_vbus_voltage (
 uint8_t port,
 PD_ADC_ID_T adc_id,
 uint8_t level)
```

This function converts the ADC level to VBUS voltage in millivolts. It takes an 8-bit ADC reading and returns the corresponding 16-bit voltage value in millivolts. This function does not perform any ADC operations. pd\_adc\_sample should be called to convert a divided version of VBus to ADC levels before calling this function.

## Parameters

|               |                                                                |
|---------------|----------------------------------------------------------------|
| <i>port</i>   | Port index. Caller should ensure to provide only valid values. |
| <i>adc_id</i> | ADC ID.                                                        |
| <i>level</i>  | The 8-bit ADC reading.                                         |

## Returns

Returns voltage in mV.

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

6.26.5.21 `pd_adc_init()`

```
ccg_status_t pd_adc_init (
 uint8_t port,
 PD_ADC_ID_T adc_id)
```

This function initializes the PD ADC block.

This function enables the PD block and the registers required for ADC operation. It then calibrates the ADC to identify the VDDD voltage. This function does not start any ADC operations.

## Parameters

|               |                                                                |
|---------------|----------------------------------------------------------------|
| <i>port</i>   | Port index. Caller should ensure to provide only valid values. |
| <i>adc_id</i> | ADC ID.                                                        |

## Returns

`ccg_status_t`

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

6.26.5.22 `pd_adc_level_to_volt()`

```
uint16_t pd_adc_level_to_volt (
 uint8_t port,
 PD_ADC_ID_T adc_id,
 uint8_t level)
```

This function converts the ADC units to voltage in millivolts.

It takes an 8-bit ADC reading and returns the corresponding 16-bit voltage value in millivolts. This function does not perform any ADC operations.

## Parameters

|               |                                                                |
|---------------|----------------------------------------------------------------|
| <i>port</i>   | Port index. Caller should ensure to provide only valid values. |
| <i>adc_id</i> | ADC ID.                                                        |
| <i>level</i>  | The 8-bit ADC reading.                                         |

**Returns**

Returns voltage in mV.

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

**6.26.5.23 pd\_adc\_sample()**

```
uint8_t pd_adc_sample (
 uint8_t port,
 PD_ADC_ID_T adc_id,
 PD_ADC_INPUT_T input)
```

This function samples the ADC.

This function enables the ADC block to function as an ADC and returns the sample value in ADC units. This function disables any previously configured comparator interrupts / settings before sampling and restores them after the sampling is done. If any interrupt scenario happens across the sampling, the information is lost.

**Parameters**

|               |                                                                |
|---------------|----------------------------------------------------------------|
| <i>port</i>   | Port index. Caller should ensure to provide only valid values. |
| <i>adc_id</i> | ADC ID.                                                        |
| <i>input</i>  | ADC input source.                                              |

**Returns**

Returns the ADC sample.

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

**6.26.5.24 pd\_adc\_select\_vref()**

```
ccg_status_t pd_adc_select_vref (
 uint8_t port,
 PD_ADC_ID_T adc_id,
 PD_ADC_VREF_T vref_sel)
```

This function selects the reference voltage used by the ADC block on the CCG device. A 2.0 V supply generated by the RefGen block in the PD IP is used as ADC reference by default. This API can be used to select the VDDD supply as the ADC reference. This is useful in cases where voltages greater than 2.0 V need to be measured.

Note: Since the VDDD supply level could vary across time, the ADC volts per division value needs to be calibrated before taking any ADC readings.

**Parameters**

|                 |                                                                |
|-----------------|----------------------------------------------------------------|
| <i>port</i>     | Port index. Caller should ensure to provide only valid values. |
| <i>adc_id</i>   | ADC ID.                                                        |
| <i>vref_sel</i> | ADC reference selection.                                       |

**Returns**

ccg\_status\_t

**Applicable devices:** CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

### 6.26.5.25 pd\_adc\_volt\_to\_level()

```
uint8_t pd_adc_volt_to_level (
 uint8_t port,
 PD_ADC_ID_T adc_id,
 uint16_t volt)
```

This function converts the voltage in millivolt to ADC units.

It takes a 16-bit voltage value in millivolts and returns the corresponding 8-bit ADC reading. This function does not perform any ADC operations.

The minimum value is limited by the PD\_ADC\_LEVEL\_MIN\_THRESHOLD value.

#### Parameters

|               |                                                                |
|---------------|----------------------------------------------------------------|
| <i>port</i>   | Port index. Caller should ensure to provide only valid values. |
| <i>adc_id</i> | ADC ID.                                                        |
| <i>volt</i>   | Voltage in mV.                                                 |

#### Returns

Returns the 8-bit ADC reading corresponding to volt.

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

### 6.26.5.26 pd\_cf\_disable()

```
void pd_cf_disable (
 uint8_t port)
```

Disables VBus Current foldback on the specified port.

#### Parameters

|             |              |
|-------------|--------------|
| <i>port</i> | USB-PD port. |
|-------------|--------------|

#### Returns

None

**Applicable devices:** CCG3PA, CCG3PA2, PAG1S.

### 6.26.5.27 pd\_cf\_enable()

```
void pd_cf_enable (
 uint8_t port,
 uint32_t cur)
```

Enables VBus Current foldback on the specified port.

#### Parameters

|             |                                    |
|-------------|------------------------------------|
| <i>port</i> | USB-PD port on which to enable CF. |
| <i>cur</i>  | Operating current in 10mA units.   |

**Returns**

None

**Applicable devices:** CCG3PA, CCG3PA2, PAG1S.**6.26.5.28 pd\_cf\_get\_status()**

```
bool pd_cf_get_status (
 uint8_t port)
```

Retrieves the status of the CF mode hardware.

**Parameters**

|             |              |
|-------------|--------------|
| <i>port</i> | USB-PD port. |
|-------------|--------------|

**Returns**

Whether the hardware module is in CF mode or not.

**Applicable devices:** CCG3PA, CCG3PA2, PAG1S.**6.26.5.29 pd\_cf\_mon\_disable()**

```
void pd_cf_mon_disable (
 uint8_t port)
```

Disables hardware monitoring for VBus Current foldback on the specified port. Monitoring should be disabled when there is VBUS transition as well as when not in PPS mode operation.

**Parameters**

|             |                                    |
|-------------|------------------------------------|
| <i>port</i> | USB-PD port on which to enable CF. |
|-------------|------------------------------------|

**Returns**

None.

**Applicable devices:** PAG1S.**6.26.5.30 pd\_cf\_mon\_enable()**

```
void pd_cf_mon_enable (
 uint8_t port,
 bool reset_state,
 vbus_cf_cbk_t cbk)
```

Enables hardware monitoring for VBus Current foldback on the specified port. The function invokes the callback when a change happens from the current state. When the monitor is enabled, the calling function can specify the default state to allow for both high to low and low to high detection at enable. This avoids any race condition while enabling the monitoring. Monitoring should be enabled only when no VBUS transition is enabled.

**Parameters**

|             |                                    |
|-------------|------------------------------------|
| <i>port</i> | USB-PD port on which to enable CF. |
|-------------|------------------------------------|

## Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <i>reset_state</i> | Current CF state as expected by the calling function. |
| <i>cbk</i>         | Function to be called when CF event is detected.      |

## Returns

None.

**Applicable devices:** PAG1S.

6.26.5.31 `pd_cmp_get_status()`

```
bool pd_cmp_get_status (
 uint8_t port,
 filter_id_t id,
 bool is_filtered)
```

Gets the current status of the VBUS / LSCSA comparators.

## Parameters

|                    |                                                                |
|--------------------|----------------------------------------------------------------|
| <i>port</i>        | USB-PD port.                                                   |
| <i>id</i>          | comparator filter ID for which the status has to be retrieved. |
| <i>is_filtered</i> | Whether to get the filtered or unfiltered status.              |

## Returns

State of the comparator.

**Applicable devices:** CCG3PA, CCG3PA2, PAG1S.

6.26.5.32 `pd_connect_vbus_div_to_amux()`

```
void pd_connect_vbus_div_to_amux (
 uint8_t port)
```

Enable connection of the internal VBus divider to AMUX bus. In case of PAG1S, this function call connects VBU↔S\_DIV to ADC AMUX. Caller is expected to ensure that the ADC is not configured for interrupt using VBUS\_MON.

## Parameters

|             |                        |
|-------------|------------------------|
| <i>port</i> | PD port to be updated. |
|-------------|------------------------|

## Returns

None

**Applicable devices:** CCG3, PAG1S.

6.26.5.33 `pd_disconnect_ra()`

```
void pd_disconnect_ra (
 void)
```

Disconnect Ra from VCONN line This function removes Ra from VCONN. This can be used by AMAs to reduce power consumption.

#### Returns

None

**Applicable devices:** CCG3, CCG3PA2.

#### 6.26.5.34 pd\_disconnect\_vbus\_div\_from\_amux()

```
bool pd_disconnect_vbus_div_from_amux (
 uint8_t port)
```

Disconnect the internal VBus divider from AMUX bus. In case of PAG1S, this function call disconnects VBUS\_DIV from ADC AMUX, and allows GLOBAL AMUX connection to ADC. Caller is expected to ensure that the ADC is not configured for interrupt using VBUS\_MON.

#### Parameters

|             |                        |
|-------------|------------------------|
| <i>port</i> | PD port to be updated. |
|-------------|------------------------|

#### Returns

true if disconnected, false if disconnection not allowed.

**Applicable devices:** CCG3, PAG1S.

#### 6.26.5.35 pd\_enable\_vconn\_comp()

```
void pd_enable_vconn_comp (
 void)
```

Enable VCONN Comparator for VCONN monitoring. This function is only applicable in VConn powered UFP applications.

#### Returns

None

**Applicable devices:** CCG3.

#### 6.26.5.36 pd\_fet\_automode\_disable()

```
void pd_fet_automode_disable (
 uint8_t port,
 bool pctrl,
 filter_id_t filter_index)
```

Disable automatic hardware control on a gate driver due to a specified comparator and output filter. Does not affect automatic control of the gate driver by other comparators of filters.

#### Parameters

|                     |                                                              |
|---------------------|--------------------------------------------------------------|
| <i>port</i>         | PD port index.                                               |
| <i>pctrl</i>        | true for provider FET driver, false for consumer FET driver. |
| <i>filter_index</i> | Filter whose control is being disabled.                      |

**Returns**

None

**Applicable devices:** CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.**6.26.5.37 pd\_fet\_automode\_enable()**

```
void pd_fet_automode_enable (
 uint8_t port,
 bool pctrl,
 filter_id_t filter_index)
```

Enable automatic hardware control on a gate driver due to a specified comparator and output filter. Does not affect automatic control of the gate driver by other comparators or filters.

**Parameters**

|                     |                                                              |
|---------------------|--------------------------------------------------------------|
| <i>port</i>         | PD port index.                                               |
| <i>pctrl</i>        | true for provider FET driver, false for consumer FET driver. |
| <i>filter_index</i> | Filter whose control is being enabled.                       |

**Returns**

None

**Applicable devices:** CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.**6.26.5.38 pd\_frs\_rx\_disable()**

```
bool pd_frs_rx_disable (
 uint8_t port)
```

This function disables the fast role swap receive functionality.

**Parameters**

|             |                                                                |
|-------------|----------------------------------------------------------------|
| <i>port</i> | Port index. Caller should ensure to provide only valid values. |
|-------------|----------------------------------------------------------------|

**Returns**

Returns true if success otherwise returns false

**Applicable devices:** CCG3, CCG4, CCG5, CCG5C, CCG6.**6.26.5.39 pd\_frs\_rx\_enable()**

```
bool pd_frs_rx_enable (
 uint8_t port)
```

This function enables the fast role swap receive functionality. Callback registered in `pd_phy_init` will be called when fast role swap signal is received.

**Parameters**

|             |                                                                |
|-------------|----------------------------------------------------------------|
| <i>port</i> | Port index. Caller should ensure to provide only valid values. |
|-------------|----------------------------------------------------------------|



**Returns**

Returns true if success, otherwise returns false

**Applicable devices:** CCG3, CCG4, CCG5, CCG5C, CCG6.

**6.26.5.40 pd\_frs\_tx\_disable()**

```
bool pd_frs_tx_disable (
 uint8_t port)
```

This function disables the fast role swap transmit functionality.

**Parameters**

|             |                                                                |
|-------------|----------------------------------------------------------------|
| <i>port</i> | Port index. Caller should ensure to provide only valid values. |
|-------------|----------------------------------------------------------------|

**Returns**

Returns true if success otherwise returns false

**Applicable devices:** CCG3.

**6.26.5.41 pd\_frs\_tx\_enable()**

```
bool pd_frs_tx_enable (
 uint8_t port)
```

This function enables the fast role swap transmit functionality. Callback registered in pd\_phy\_init will be called when fast role swap signal is transmitted.

**Parameters**

|             |                                                                |
|-------------|----------------------------------------------------------------|
| <i>port</i> | Port index. Caller should ensure to provide only valid values. |
|-------------|----------------------------------------------------------------|

**Returns**

Returns true if success otherwise returns false

**Applicable devices:** CCG3.

**6.26.5.42 pd\_get\_vbus\_adc\_level()**

```
uint8_t pd_get_vbus_adc_level (
 uint8_t port,
 PD_ADC_ID_T adc_id,
 uint16_t volt,
 int8_t per)
```

This function gets the ADC level that corresponds to the actual voltage on vbus. It also takes into account the VBus monitor divider.

**Parameters**

|               |                                                                |
|---------------|----------------------------------------------------------------|
| <i>port</i>   | Port index. Caller should ensure to provide only valid values. |
| <i>adc_id</i> | ADC ID.                                                        |

**Parameters**

|             |                                   |
|-------------|-----------------------------------|
| <i>volt</i> | Voltage in 50mV units.            |
| <i>per</i>  | Percentage margin on the voltage. |

**Returns**

Returns the ADC level.

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

**6.26.5.43 pd\_get\_vconn\_status()**

```
bool pd_get_vconn_status (
 void)
```

Get current status of VCONN supply. This function checks if port partner is providing VCONN and returns the status. Expectation is that VCONN comparator is enabled first using `pd_enable_vconn_comp` function before calling this function. This function is expected to be used by VCONN powered accessories.

**Returns**

true if VCONN is present, false otherwise

**Applicable devices:** CCG3.

**6.26.5.44 pd\_hal\_abort\_auto\_toggle()**

```
void pd_hal_abort_auto_toggle (
 uint8_t port)
```

Abort any ongoing automatic DRP toggle operation.

**Parameters**

|             |                                        |
|-------------|----------------------------------------|
| <i>port</i> | Port on which toggle is to be aborted. |
|-------------|----------------------------------------|

**Returns**

None

**Applicable devices:** CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6.

**6.26.5.45 pd\_hal\_cleanup()**

```
void pd_hal_cleanup (
 uint8_t port)
```

This function cleans up the PD block after a disconnect.

**Parameters**

|             |                                                                |
|-------------|----------------------------------------------------------------|
| <i>port</i> | Port index. Caller should ensure to provide only valid values. |
|-------------|----------------------------------------------------------------|

**Returns**

None.

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.**6.26.5.46 pd\_hal\_config\_auto\_toggle()**

```
void pd_hal_config_auto_toggle (
 uint8_t port,
 bool enable)
```

Enable/disable automatic hardware toggle operation as part of deep sleep cycle.

**Parameters**

|               |                                            |
|---------------|--------------------------------------------|
| <i>port</i>   | PD port to be updated.                     |
| <i>enable</i> | Whether automatic toggle is to be enabled. |

**Returns**

None

**Applicable devices:** CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6.**6.26.5.47 pd\_hal\_dual\_fet\_config()**

```
void pd_hal_dual_fet_config (
 bool dual_fet,
 uint8_t spacing)
```

Configures FET parameters for CCG3 devices.

This function configures various FET control parameters for the device. It should be called before the PD HAL is initialized. Also it should be called only once during initialization. Since this is high voltage FET configuration, any wrong configuration shall result in damage of the device and boards. The function should be called only when the default behaviour needs to be overridden.

Caller should ensure that the FET access is not done before it is configured and is never configured while the PD stack is in operation.

CCG3 devices shall use dual FET configuration with spacing of 10LF cycles by default. If this function is not invoked, then the default configuration shall be used. CCG3 shall always function in dual FET mode as the FET controls are dedicated IOs.

**Parameters**

|                 |                                                                                                                                                                                                                |
|-----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>dual_fet</i> | Set to true if the system uses dual FETs for each direction. Otherwise set to false.                                                                                                                           |
| <i>spacing</i>  | Spacing in LF-cycles between dual FETs for firmware based turn-on and turn-off. Auto shut-off and turn-on happens simultaneously. Valid only for dual_fet configuration. Set to zero for simultaneous control. |

**Returns**

None

**Warning**

Misconfiguration of this parameter shall result in board damage.

**Applicable devices:** CCG3.

**6.26.5.48 pd\_hal\_enable\_internal\_vbus\_mon()**

```
void pd_hal_enable_internal_vbus_mon (
 bool enable)
```

Enable/disable the internal VBus monitor function in the CCGx device.

**Parameters**

|               |                                          |
|---------------|------------------------------------------|
| <i>enable</i> | Whether to enable internal VBus Monitor. |
|---------------|------------------------------------------|

**Returns**

None

**Applicable devices:** CCG3, CCG4.

**6.26.5.49 pd\_hal\_get\_vbus\_csa\_rsense()**

```
uint8_t pd_hal_get_vbus_csa_rsense (
 void)
```

Returns the RSENSE value used for CSA operation. The function is valid only for CCG3PA/CCG3PA2 implementation currently. The value specified is in 100uOhm units.

**Returns**

The total Rsense in 100uOhm units.

**Applicable devices:** CCG3PA, CCG3PA2, PAG1S.

**6.26.5.50 pd\_hal\_get\_vbus\_detach\_adc()**

```
PD_ADC_ID_T pd_hal_get_vbus_detach_adc (
 void)
```

Identify the CCGx ADC that is used for VBus detach detection.

**Returns**

ID of the ADC block used for VBus detach detection.

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

**6.26.5.51 pd\_hal\_get\_vbus\_detach\_input()**

```
PD_ADC_INPUT_T pd_hal_get_vbus_detach_input (
 void)
```

Identify the ADC input that is used for VBus detach detection.

**Returns**

ADC block input used for VBus detach detection.

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

## 6.26.5.52 pd\_hal\_init()

```
ccg_status_t pd_hal_init (
 uint8_t port)
```

This function initializes the PDSS IP with necessary clock and interrupt handlers.

**Parameters**

|             |                                                                |
|-------------|----------------------------------------------------------------|
| <i>port</i> | Port index. Caller should ensure to provide only valid values. |
|-------------|----------------------------------------------------------------|

**Returns**

ccg\_status\_t.

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

## 6.26.5.53 pd\_hal\_is\_auto\_toggle\_active()

```
bool pd_hal_is_auto_toggle_active (
 uint8_t port)
```

Check whether automatic DRP toggle operation is active.

**Parameters**

|             |                        |
|-------------|------------------------|
| <i>port</i> | PD port to be checked. |
|-------------|------------------------|

**Returns**

Current status of automatic DRP toggle operation.

**Applicable devices:** CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6.

## 6.26.5.54 pd\_hal\_is\_sink\_fet\_on()

```
bool pd_hal_is_sink_fet_on (
 uint8_t port)
```

Check whether the Sink FET has been turned on.

**Parameters**

|             |                |
|-------------|----------------|
| <i>port</i> | PD port index. |
|-------------|----------------|

**Returns**

true if the sink path is on to allow system to charge.

**Applicable devices:** CCG5, CCG5C, CCG6

#### 6.26.5.55 pd\_hal\_measure\_ea\_volt()

```
uint16_t pd_hal_measure_ea_volt (
 uint8_t port)
```

Measure the EA voltage. The function saturates at 2V.

##### Parameters

|             |              |
|-------------|--------------|
| <i>port</i> | USB-PD port. |
|-------------|--------------|

##### Returns

EA voltage in mV.

**Applicable devices:** PAG1S.

#### 6.26.5.56 pd\_hal\_measure\_line\_volt()

```
uint16_t pd_hal_measure_line_volt (
 uint8_t port)
```

Measure the line voltage equivalent via line feedforward.

##### Parameters

|             |              |
|-------------|--------------|
| <i>port</i> | USB-PD port. |
|-------------|--------------|

##### Returns

Line voltage in volts.

**Applicable devices:** PAG1S.

#### 6.26.5.57 pd\_hal\_measure\_vbus()

```
uint16_t pd_hal_measure_vbus (
 uint8_t port)
```

Measure the current voltage on the VBus supply.

##### Parameters

|             |                         |
|-------------|-------------------------|
| <i>port</i> | PD port to be measured. |
|-------------|-------------------------|

##### Returns

VBus voltage in mV units.

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

#### 6.26.5.58 pd\_hal\_measure\_vbus\_cur()

```
uint16_t pd_hal_measure_vbus_cur (
 uint8_t port)
```

Measure the current currently being provided through the VBus supply.

#### Parameters

|             |                                             |
|-------------|---------------------------------------------|
| <i>port</i> | PD port on which current is to be measured. |
|-------------|---------------------------------------------|

#### Returns

VBus active current in 10 mA units.

**Applicable devices:** CCG3PA, CCG3PA2, CCG6, PAG1S.

#### 6.26.5.59 pd\_hal\_measure\_vbus\_in()

```
uint16_t pd_hal_measure_vbus_in (
 uint8_t port)
```

Measure the current voltage on the VBus-in.

#### Parameters

|             |                         |
|-------------|-------------------------|
| <i>port</i> | PD port to be measured. |
|-------------|-------------------------|

#### Returns

VBus-in voltage in mV units.

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

#### 6.26.5.60 pd\_hal\_set\_cc\_ovp\_pending()

```
void pd_hal_set_cc_ovp_pending (
 uint8_t port)
```

Notify the HAL that an OVP condition is pending on the CC line. This will update the logic used to poll CC status in the HAL.

#### Parameters

|             |                                          |
|-------------|------------------------------------------|
| <i>port</i> | PD port index on which fault is pending. |
|-------------|------------------------------------------|

#### Returns

None

**Applicable devices:** CCG5, CCG5C, CCG6.

#### 6.26.5.61 pd\_hal\_set\_fet\_drive()

```
void pd_hal_set_fet_drive (
 pd_fet_dr_t pctrl_drive,
 pd_fet_dr_t cctrl_drive)
```

Configures the drive modes for the FETs.

This function allows the application to select polarity of drive for CCG4 devices and N-FET drive or P-FET drive for CCG3 devices. The configuration should match the hardware implementation on the board.

CCG3 devices support N-FET by default for both PCTRL and CCTRL. Override this only for the boards with P-FETs. Standard Cypress reference schematics and kits use N-FETs and changing this shall result in board damage.

CCG4 devices support active high polarity for PCTRL (source) and active low polarity for CCTRL (sink) by default. This configuration matches the standard Cypress reference schematics and kits and changing this shall result in board damage.

This function is expected to be called once, before the `pdss_hal` is initialized. Calls during stack operation are not restricted but are likely to result in spurious behaviour and / or board damage.

#### Parameters

|                    |                                                           |
|--------------------|-----------------------------------------------------------|
| <i>pctrl_drive</i> | Select the type of gate driver used for the Provider FET. |
| <i>cctrl_drive</i> | Select the type of gate driver used for the Consumer FET. |

#### Returns

None

#### Warning

Misconfiguration of this parameter will result in board damage.

**Applicable devices:** CCG3, CCG4.

#### 6.26.5.62 `pd_hal_set_reference()`

```
void pd_hal_set_reference (
 uint8_t port,
 bool flag)
```

Facilitates to set deep sleep/bandgap reference. Function should be called for setting deep sleep reference just before entry into deep sleep state. Function should be called with band gap reference just after exit from deep sleep state.

#### Parameters

|             |                                                       |
|-------------|-------------------------------------------------------|
| <i>port</i> | PD port to change reference.                          |
| <i>flag</i> | If true signifies deep sleep reference, else band gap |

#### Returns

None.

**Applicable devices:** CCG3PA, CCG3PA2.

#### 6.26.5.63 `pd_hal_set_supply_change_evt_cb()`

```
void pd_hal_set_supply_change_evt_cb (
 pd_supply_change_cbk_t cb)
```

Register a callback that can be used for notification of power supply changes.



## Parameters

|           |                            |
|-----------|----------------------------|
| <i>cb</i> | Callback function pointer. |
|-----------|----------------------------|

## Returns

None

**Applicable devices:** CCG5, CCG5C, CCG6.

## 6.26.5.64 pd\_hal\_set\_vbus\_csa\_rsense()

```
void pd_hal_set_vbus_csa_rsense (
 uint8_t rsense)
```

Specify the RSENSE used for CSA operation. The function is valid only for CCG3PA/CCG3PA2 implementation currently. The value specified is in 100uOhm units and should include the parasitic resistances as well.

## Parameters

|               |                                    |
|---------------|------------------------------------|
| <i>rsense</i> | The total Rsense in 100uOhm units. |
|---------------|------------------------------------|

## Returns

None

**Applicable devices:** CCG3PA, CCG3PA2, PAG1S.

## 6.26.5.65 pd\_hal\_set\_vbus\_detach\_params()

```
void pd_hal_set_vbus_detach_params (
 PD_ADC_ID_T adc_id,
 PD_ADC_INPUT_T adc_inp)
```

Select the comparator block and input setting used for VBus detach detection. CCG will use the selected settings to detect Type-C disconnection based on removal of the VBus power.

## Parameters

|                |                                               |
|----------------|-----------------------------------------------|
| <i>adc_id</i>  | Select the comparator (ADC) block to be used. |
| <i>adc_inp</i> | Select the comparator input to be used.       |

## Returns

None

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

## 6.26.5.66 pd\_hal\_set\_vbus\_mon\_divider()

```
void pd_hal_set_vbus_mon_divider (
 uint8_t divider)
```

Specify the voltage division ratio between the voltage applied on VBUS\_MON and the actual VBus voltage. The commonly used resistor divider ratio used is 1:10, giving a voltage division ratio of 1/11.

## Parameters

|                |                                                  |
|----------------|--------------------------------------------------|
| <i>divider</i> | Ratio between VBUS_MON voltage and VBus voltage. |
|----------------|--------------------------------------------------|

## Returns

None

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.6.26.5.67 `pd_hal_typec_sm_restart()`

```
void pd_hal_typec_sm_restart (
 uint8_t port)
```

Restart Type-C state machine once auto toggle operation is complete. This is normally required when the port has just exited the auto DRP toggle stage, and the state machine needs to handle further operations.

## Parameters

|             |                                                  |
|-------------|--------------------------------------------------|
| <i>port</i> | Port on which state machine restart is required. |
|-------------|--------------------------------------------------|

## Returns

None

**Applicable devices:** CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.6.26.5.68 `pd_hal_vconn_ocp_disable()`

```
void pd_hal_vconn_ocp_disable (
 uint8_t port)
```

Disable Over-Current detection on the VConn power source.

## Parameters

|             |                           |
|-------------|---------------------------|
| <i>port</i> | PD port to be configured. |
|-------------|---------------------------|

## Returns

None

**Applicable devices:** CCG3, CCG4, CCG5, CCG5C, CCG6.6.26.5.69 `pd_hal_vconn_ocp_enable()`

```
void pd_hal_vconn_ocp_enable (
 uint8_t port,
 uint8_t debounce,
 PD_ADC_CB_T cb)
```

Enable Over-Current detection on the VConn power source.

## Parameters

|                 |                                                    |
|-----------------|----------------------------------------------------|
| <i>port</i>     | PD port to be configured.                          |
| <i>debounce</i> | Debounce period in milliseconds.                   |
| <i>cb</i>       | Callback function to be called on fault detection. |

## Returns

None

**Applicable devices:** CCG3, CCG4, CCG5, CCG5C, CCG6.

## 6.26.5.70 pd\_internal\_cfet\_off()

```
void pd_internal_cfet_off (
 uint8_t port,
 bool turn_off_seq)
```

Turn off consumer FET using the internal gate driver.

## Parameters

|                     |                                                                                                                                                                         |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>port</i>         | Port index. Caller should ensure to provide only valid values.                                                                                                          |
| <i>turn_off_seq</i> | On CCG3, this bit selects which FET turns off first. <ul style="list-style-type: none"> <li>• 0: FET 0 turns off first.</li> <li>• 1: FET 1 turns off first.</li> </ul> |

## Returns

None

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

## 6.26.5.71 pd\_internal\_cfet\_on()

```
void pd_internal_cfet_on (
 uint8_t port,
 bool turn_on_seq)
```

Turn on consumer FET using the internal gate driver.

## Parameters

|                    |                                                                                                                                                                      |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>port</i>        | Port index. Caller should ensure to provide only valid values.                                                                                                       |
| <i>turn_on_seq</i> | On CCG3, this bit selects which FET turns on first. <ul style="list-style-type: none"> <li>• 0: FET 0 turns on first.</li> <li>• 1: FET 1 turns on first.</li> </ul> |

**Returns**

Void.

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

**6.26.5.72 pd\_internal\_cfet\_soft\_start\_off()**

```
void pd_internal_cfet_soft_start_off (
 uint8_t port)
```

Resets the soft state state machine associated with the Consumer FET gate driver. This needs to be called whenever the FET is being disabled so that soft start is used during the next enable sequence.

**Parameters**

|             |                |
|-------------|----------------|
| <i>port</i> | PD port index. |
|-------------|----------------|

**Returns**

None

**Applicable devices:** CCG3PA, CCG3PA2, CCG6.

**6.26.5.73 pd\_internal\_cfet\_soft\_start\_on()**

```
void pd_internal_cfet_soft_start_on (
 uint8_t port,
 uint8_t fet_status)
```

Initiate the soft enable of the Consumer FET to control inrush current. This function is called as part of the `pd↔_internal_cfet_on` call and slowly increases the gate driver pull-down current strength until the FET is fully turned ON.

**Parameters**

|                   |                                                                                                                                                                  |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>port</i>       | PD port index.                                                                                                                                                   |
| <i>fet_status</i> | Flag indicating whether the gate driver is already in the enabled state. Used in cases where there are back-to-back <a href="#">pd_internal_cfet_on()</a> calls. |

**Returns**

None

**Applicable devices:** CCG3PA, CCG3PA2, CCG6.

**6.26.5.74 pd\_internal\_pfet\_off()**

```
void pd_internal_pfet_off (
 uint8_t port,
 bool turn_off_seq)
```

Turn off producer FET using the internal gate driver.

**Parameters**

|             |                                                                |
|-------------|----------------------------------------------------------------|
| <i>port</i> | Port index. Caller should ensure to provide only valid values. |
|-------------|----------------------------------------------------------------|

## Parameters

|                     |                                                                                                                                                                         |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>turn_off_seq</i> | On CCG3, this bit selects which FET turns off first. <ul style="list-style-type: none"> <li>• 0: FET 0 turns off first.</li> <li>• 1: FET 1 turns off first.</li> </ul> |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Returns

Void.

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

## 6.26.5.75 pd\_internal\_pfet\_on()

```
void pd_internal_pfet_on (
 uint8_t port,
 bool turn_on_seq)
```

Turn on producer FET using the internal gate driver or dedicated GPIO.

## Parameters

|                    |                                                                                                                                                                      |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>port</i>        | Port index. Caller should ensure to provide only valid values.                                                                                                       |
| <i>turn_on_seq</i> | On CCG3, this bit selects which FET turns on first. <ul style="list-style-type: none"> <li>• 0: FET 0 turns on first.</li> <li>• 1: FET 1 turns on first.</li> </ul> |

## Returns

None

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

## 6.26.5.76 pd\_internal\_pfet\_soft\_start\_off()

```
void pd_internal_pfet_soft_start_off (
 uint8_t port)
```

Resets the soft state state machine associated with the Provider FET gate driver. This needs to be called whenever the FET is being disabled so that soft start is used during the next enable sequence.

## Parameters

|             |                |
|-------------|----------------|
| <i>port</i> | PD port index. |
|-------------|----------------|

## Returns

None

**Applicable devices:** CCG3PA, CCG3PA2, CCG6.

### 6.26.5.77 pd\_internal\_pfet\_soft\_start\_on()

```
void pd_internal_pfet_soft_start_on (
 uint8_t port,
 uint8_t fet_status)
```

Initiate the soft enable of the Provider FET to control inrush current. This function is called as part of the `pd_internal_pfet_on` call and slowly increases the gate driver pull-down current strength until the FET is fully turned ON.

#### Parameters

|                   |                                                                                                                                                               |
|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>port</i>       | PD port index.                                                                                                                                                |
| <i>fet_status</i> | Flag indicating whether the gate driver is already in the enabled state. Used in cases where there are back-to-back <code>pd_internal_pfet_on()</code> calls. |

#### Returns

None

**Applicable devices:** CCG3PA, CCG3PA2, CCG6.

### 6.26.5.78 pd\_internal\_vbus\_discharge\_off()

```
void pd_internal_vbus_discharge_off (
 uint8_t port)
```

Turn off the internal VBus discharge path.

#### Parameters

|             |                                         |
|-------------|-----------------------------------------|
| <i>port</i> | Port on which to enable VBus discharge. |
|-------------|-----------------------------------------|

#### Returns

None

**Applicable devices:** CCG3, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

### 6.26.5.79 pd\_internal\_vbus\_discharge\_on()

```
void pd_internal_vbus_discharge_on (
 uint8_t port)
```

Turn on the internal VBus discharge path.

#### Parameters

|             |                                         |
|-------------|-----------------------------------------|
| <i>port</i> | Port on which to enable VBus discharge. |
|-------------|-----------------------------------------|

#### Returns

None

**Applicable devices:** CCG3, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

## 6.26.5.80 pd\_internal\_vbus\_in\_discharge\_off()

```
void pd_internal_vbus_in_discharge_off (
 uint8_t port)
```

Turn off VBUS\_IN discharge path.

## Parameters

|             |                                            |
|-------------|--------------------------------------------|
| <i>port</i> | Port on which to enable VBUS_IN discharge. |
|-------------|--------------------------------------------|

## Returns

None

**Applicable devices:** CCG3PA, CCG3PA2, PAG1S.

## 6.26.5.81 pd\_internal\_vbus\_in\_discharge\_on()

```
void pd_internal_vbus_in_discharge_on (
 uint8_t port)
```

Turn on the internal VBUS\_IN discharge path.

## Parameters

|             |                                            |
|-------------|--------------------------------------------|
| <i>port</i> | Port on which to enable VBUS_IN discharge. |
|-------------|--------------------------------------------|

## Returns

None

**Applicable devices:** CCG3PA, CCG3PA2, PAG1S.

## 6.26.5.82 pd\_internal\_vbus\_load\_change\_isr\_enable()

```
void pd_internal_vbus_load_change_isr_enable (
 uint8_t port,
 uint32_t cur,
 uint8_t filter,
 vbus_load_chg_cbk_t cbk)
```

Enable load change monitor ISR capability.

The function allows multiplexing the internal OCP comparator to perform low to high load transition on VBUS along with the regular OCP functionality.

## Parameters

|               |                                                            |
|---------------|------------------------------------------------------------|
| <i>port</i>   | PD port index.                                             |
| <i>cur</i>    | VBUS current threshold in 10mA units to trigger ISR        |
| <i>filter</i> | Filter settings in filter clock cycles to be used.         |
| <i>cbk</i>    | Callback function to be invoked when a change is detected. |

**Returns**

None

**Applicable devices:** CCG3PA, CCG3PA2, PAG1S.**6.26.5.83 pd\_internal\_vbus\_ocp\_dis()**

```
void pd_internal_vbus_ocp_dis (
 uint8_t port,
 bool pctrl)
```

Disable Over Current Protection (OCP) control on the PD port.

**Parameters**

|              |                                                                       |
|--------------|-----------------------------------------------------------------------|
| <i>port</i>  | PD port index.                                                        |
| <i>pctrl</i> | Type of gate driver to be turned off on fault. Should be set to true. |

**Returns**

None

**Applicable devices:** CCG3, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.**6.26.5.84 pd\_internal\_vbus\_ocp\_en()**

```
void pd_internal_vbus_ocp_en (
 uint8_t port,
 uint8_t av_bw,
 uint8_t vref_sel,
 bool pctrl,
 uint8_t mode,
 uint8_t debounce_ms)
```

Enable Over Current Protection (OCP) control using the internal Current Sense Amplifier.

**Parameters**

|                    |                                                                       |
|--------------------|-----------------------------------------------------------------------|
| <i>port</i>        | PD port index.                                                        |
| <i>av_bw</i>       | Gain selection for the Current Sense Amplifier.                       |
| <i>vref_sel</i>    | Reference voltage used to detect Over-Current condition.              |
| <i>pctrl</i>       | Type of gate driver to be turned off on fault. Should be set to true. |
| <i>mode</i>        | Mode of OCP fault handling.                                           |
| <i>debounce_ms</i> | OCP software debounce timeout in ms.                                  |

**Returns**

None

**Applicable devices:** CCG3, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.**6.26.5.85 pd\_internal\_vbus\_ovp\_dis()**

```
void pd_internal_vbus_ovp_dis (
```



```
uint8_t port,
bool pctrl)
```

Disable Over Voltage Protection (OVP) control.

#### Parameters

|              |                                                                                                 |
|--------------|-------------------------------------------------------------------------------------------------|
| <i>port</i>  | PD port index.                                                                                  |
| <i>pctrl</i> | Flag indicating the type of gate driver to be controlled, true for P_CTRL and false for C_CTRL. |

#### Returns

None

**Applicable devices:** CCG3, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

#### 6.26.5.86 pd\_internal\_vbus\_ovp\_en()

```
void pd_internal_vbus_ovp_en (
 uint8_t port,
 uint16_t volt,
 int8_t per,
 PD_ADC_CB_T cb,
 bool pctrl,
 vbus_ovp_mode_t mode,
 uint8_t filter_sel)
```

Enable Over Voltage Protection (OVP) control using the internal UV-OV block.

#### Parameters

|                   |                                                                                                 |
|-------------------|-------------------------------------------------------------------------------------------------|
| <i>port</i>       | PD port index.                                                                                  |
| <i>volt</i>       | Contract Voltage in mV units.                                                                   |
| <i>per</i>        | Percentage margin on the contract voltage.                                                      |
| <i>cb</i>         | Callback to be called on fault detection.                                                       |
| <i>pctrl</i>      | Flag indicating the type of gate driver to be controlled, true for P_CTRL and false for C_CTRL. |
| <i>mode</i>       | OVP mode selection.                                                                             |
| <i>filter_sel</i> | The delay in filter clock cycles between the OVP detection and trigger.                         |

#### Returns

None

**Applicable devices:** CCG3, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

#### 6.26.5.87 pd\_internal\_vbus\_rcp\_dis()

```
void pd_internal_vbus_rcp_dis (
 uint8_t port,
 bool pctrl)
```

Disable Reverse Current Protection (RCP) control using internal Current Sense Amplifier. All forms of RCP detection will be disabled.

## Parameters

|              |                                                                                         |
|--------------|-----------------------------------------------------------------------------------------|
| <i>port</i>  | PD port index.                                                                          |
| <i>pctrl</i> | Flag indicating the type of gate driver to be controlled on RCP. Should be set to true. |

## Returns

None

**Applicable devices:** CCG6.6.26.5.88 `pd_internal_vbus_rcp_en()`

```
void pd_internal_vbus_rcp_en (
 uint8_t port,
 bool pctrl,
 uint8_t csa_det_en,
 uint8_t cmp_det_en,
 uint8_t ovp_det_en)
```

Enable Reverse Current Protection (RCP) control using internal Current Sense Amplifier. RCP can only be enabled when CCG is the power source. Since RCP response needs to be fast, the function only supports an automatic FET turn-off mode of handling.

## Parameters

|                   |                                                                                         |
|-------------------|-----------------------------------------------------------------------------------------|
| <i>port</i>       | PD port index.                                                                          |
| <i>pctrl</i>      | Flag indicating the type of gate driver to be controlled on RCP. Should be set to true. |
| <i>csa_det_en</i> | Enable RCP detection by measuring voltage drop between CSN and CSP pins.                |
| <i>cmp_det_en</i> | Enable RCP detection by measuring voltage drop between VBus and CSN pins.               |
| <i>ovp_det_en</i> | Enable RCP detection by measuring voltage on the CSN pin.                               |

## Returns

Null

**Applicable devices:** CCG6.6.26.5.89 `pd_internal_vbus_scp_dis()`

```
void pd_internal_vbus_scp_dis (
 uint8_t port,
 bool pctrl)
```

Disable Short Circuit Protection (SCP) control.

## Parameters

|              |                                                                                         |
|--------------|-----------------------------------------------------------------------------------------|
| <i>port</i>  | PD port index.                                                                          |
| <i>pctrl</i> | Flag indicating the type of gate driver to be controlled on SCP. Should be set to true. |

## Returns

NULL

**Applicable devices:** CCG3PA, CCG3PA2, CCG6, PAG1S.

## 6.26.5.90 pd\_internal\_vbus\_scp\_en()

```
void pd_internal_vbus_scp_en (
 uint8_t port,
 uint32_t vsense,
 uint8_t filter_sel,
 bool pctrl,
 uint8_t mode)
```

Enable Short Circuit Protection (SCP) control using the internal Current Sense Amplifier. SCP can only be enabled when CCG is the power source.

## Parameters

|                   |                                                                                                                                                                                                   |
|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>port</i>       | PD port index.                                                                                                                                                                                    |
| <i>vsense</i>     | The reference voltage that should be compared against the output of the Current Sense Amplifier to detect SCP. This value is dependent on the max. current, the sense impedance and the CSA gain. |
| <i>filter_sel</i> | The delay in filter clock cycles between the SCP detection and trigger.                                                                                                                           |
| <i>pctrl</i>      | Flag indicating the type of gate driver to be controlled on SCP. Should be set to true.                                                                                                           |
| <i>mode</i>       | SCP fault handling mode selection.                                                                                                                                                                |

## Returns

Null

**Applicable devices:** CCG3PA, CCG3PA2, CCG6, PAG1S.

## 6.26.5.91 pd\_internal\_vbus\_uvp\_dis()

```
void pd_internal_vbus_uvp_dis (
 uint8_t port,
 bool pctrl)
```

Disable Under Voltage Protection (UVP) control.

## Parameters

|              |                                                                                                 |
|--------------|-------------------------------------------------------------------------------------------------|
| <i>port</i>  | PD port index.                                                                                  |
| <i>pctrl</i> | Flag indicating the type of gate driver to be controlled, true for P_CTRL and false for C_CTRL. |

## Returns

Null.

**Applicable devices:** CCG3, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

## 6.26.5.92 pd\_internal\_vbus\_uvp\_en()

```
void pd_internal_vbus_uvp_en (
```

```

uint8_t port,
uint16_t volt,
int8_t per,
PD_ADC_CB_T cb,
bool pctrl,
vbus_uvp_mode_t mode,
uint8_t filter_sel)

```

Enable Under Voltage Protection (UVP) control using the internal UV-OV block. UVP is only expected to be used while CCG is the power source.

#### Parameters

|                   |                                                                                                 |
|-------------------|-------------------------------------------------------------------------------------------------|
| <i>port</i>       | PD port index.                                                                                  |
| <i>volt</i>       | Minimum contract voltage in mV units.                                                           |
| <i>per</i>        | Percentage of voltage drop that is acceptable.                                                  |
| <i>cb</i>         | Callback to be called on fault detection.                                                       |
| <i>pctrl</i>      | Flag indicating the type of gate driver to be controlled, true for P_CTRL and false for C_CTRL. |
| <i>mode</i>       | UVP handling mode selection.                                                                    |
| <i>filter_sel</i> | The delay in filter clock cycles between the UVP detection and trigger.                         |

#### Returns

Null.

**Applicable devices:** CCG3, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

#### 6.26.5.93 pd\_is\_v5v\_supply\_on()

```

bool pd_is_v5v_supply_on (
 uint8_t port)

```

Check whether the 5V supply to provide VConn power is present.

#### Parameters

|             |             |
|-------------|-------------|
| <i>port</i> | Port index. |
|-------------|-------------|

#### Returns

true if 5V supply is present.

**Applicable devices:** CCG5, CCG5C, CCG6.

#### 6.26.5.94 pd\_is\_vconn\_present()

```

bool pd_is_vconn_present (
 uint8_t port,
 uint8_t channel)

```

This function gets Vconn status for the specified channel.

#### Parameters

|                |                                                                |
|----------------|----------------------------------------------------------------|
| <i>port</i>    | Port index. Caller should ensure to provide only valid values. |
| <i>channel</i> | Channel index, where CC1 = 0, CC2 = 1.                         |

**Returns**

Returns true if Vconn is turned on, false otherwise.

**Applicable devices:** CCG3, CCG4, CCG5, CCG5C, CCG6.

**6.26.5.95 pd\_lscsa\_calc\_cfg()**

```
void pd_lscsa_calc_cfg (
 uint32_t cur_10mA,
 uint8_t * gain_sel,
 uint8_t * vref_sel)
```

Calculates the gain and reference settings for a specific current requirement. This function eliminates the precalculated table.

**Parameters**

|                 |                                                                                                               |
|-----------------|---------------------------------------------------------------------------------------------------------------|
| <i>cur_10mA</i> | Current setting in 10mA unit for which CSA setting is required.                                               |
| <i>gain_sel</i> | Pointer to variable where the gain setting code corresponding to the current setting needs to be stored.      |
| <i>vref_sel</i> | Pointer to variable where the voltage reference code corresponding to the current setting needs to be stored. |

**Returns**

None

**Applicable devices:** CCG3PA, CCG3PA2, PAG1S.

**6.26.5.96 pd\_lscsa\_cfg()**

```
ccg_status_t pd_lscsa_cfg (
 lscsa_app_config_t lscsa_app,
 uint8_t gain_sel)
```

Sets up LSCSA for specified application.

**Parameters**

|                  |                                |
|------------------|--------------------------------|
| <i>lscsa_app</i> | LSCSA application              |
| <i>gain_sel</i>  | Gain selection code to be used |

**Returns**

ccg\_status\_t

**Applicable devices:** CCG3PA, CCG3PA2, PAG1S.

**6.26.5.97 pd\_pasc\_get\_valley()**

```
uint8_t pd_pasc_get_valley (
 uint8_t port)
```

Get the current valley for PWM operation.

**Parameters**

|             |              |
|-------------|--------------|
| <i>port</i> | USB-PD port. |
|-------------|--------------|

**Returns**

Current operating valley.

**Applicable devices:** PAG1S.

**6.26.5.98 pd\_pasc\_lp\_disable()**

```
void pd_pasc_lp_disable (
 uint8_t port)
```

Disable the low power operation for the regulation.

This function should be called only from the early attach handler. Calling this from anywhere else can result in unstable system. This is used when system needs to be in low power mode operation.

**Parameters**

|             |              |
|-------------|--------------|
| <i>port</i> | USB-PD port. |
|-------------|--------------|

**Applicable devices:** PAG1S.

**6.26.5.99 pd\_pasc\_lp\_ds\_allowed()**

```
bool pd_pasc_lp_ds_allowed (
 uint8_t port)
```

Returns whether system regulator low power mode operation is ready or not.

This function should be called from the sleep / deep sleep check to start the deep sleep mode operation.

**Parameters**

|             |              |
|-------------|--------------|
| <i>port</i> | USB-PD port. |
|-------------|--------------|

**Returns**

true=If LP mode is ready; false=if not ready.

**Applicable devices:** PAG1S.

**6.26.5.100 pd\_pasc\_lp\_enable()**

```
void pd_pasc_lp_enable (
 uint8_t port)
```

Enable the low power operation for the regulation.

This function should be called only from the detach handler. Calling this from anywhere else can result in unstable system. This is used when system needs to be in low power mode operation.

## Parameters

|             |              |
|-------------|--------------|
| <i>port</i> | USB-PD port. |
|-------------|--------------|

**Applicable devices:** PAG1S.

## 6.26.5.101 pd\_pasc\_lp\_is\_active()

```
bool pd_pasc_lp_is_active (
 uint8_t port)
```

Returns whether system regulator low power mode operation is active or not.

This function should be called from the sleep / deep sleep check to start the deep sleep mode operation.

## Parameters

|             |              |
|-------------|--------------|
| <i>port</i> | USB-PD port. |
|-------------|--------------|

## Returns

true=If LP mode is active; false=if not enabled.

**Applicable devices:** PAG1S.

## 6.26.5.102 pd\_pasc\_lp\_task()

```
void pd_pasc_lp_task (
 uint8_t port)
```

Low power task handler for regulation.

This function should be called only from the main task loop. Calling this from anywhere else can result in unstable system. This is used when system needs to be in low power mode operation.

## Parameters

|             |              |
|-------------|--------------|
| <i>port</i> | USB-PD port. |
|-------------|--------------|

**Applicable devices:** PAG1S.

## 6.26.5.103 pd\_pasc\_poll\_task()

```
void pd_pasc_poll_task (
 uint8_t port)
```

Run the PWM based tasks for regulation control.

## Parameters

|             |              |
|-------------|--------------|
| <i>port</i> | USB-PD port. |
|-------------|--------------|

**Returns**

None.

**Applicable devices:** PAG1S.

**6.26.5.104 pd\_pasc\_ptdrv\_cont\_det\_disable()**

```
void pd_pasc_ptdrv_cont_det_disable (
 uint8_t port)
```

Disable contention detection interrupt in SSC mode operation.

**Parameters**

|             |              |
|-------------|--------------|
| <i>port</i> | USB-PD port. |
|-------------|--------------|

**Returns**

Status of the update request.

**Applicable devices:** PAG1S.

**6.26.5.105 pd\_pasc\_ptdrv\_cont\_det\_enable()**

```
void pd_pasc_ptdrv_cont_det_enable (
 uint8_t port,
 pasc_ptdrv_cont_cbk_t cbk)
```

Enable contention detection interrupt in SSC mode operation.

The function enables contention detection between PWM signal and SR to avoid cross conduction during sudden load transitions.

**Parameters**

|             |                            |
|-------------|----------------------------|
| <i>port</i> | USB-PD port.               |
| <i>cbk</i>  | Callback function pointer. |

**Returns**

Status of the update request.

**Applicable devices:** PAG1S.

**6.26.5.106 pd\_pasc\_send\_stop\_signal()**

```
void pd_pasc_send_stop_signal (
 uint8_t port)
```

Send stop signal to PAG1P device.

This function should be called only when a fatal regulation error is detected. The function sends stop signal to the PAG1P device in SSC mode operation. PAG1P on receiving this signal shall stop pulsing the primary side. This shall result in VBUS\_IN going to zero. The only recovery path for this is AC power cycle at the primary end.



## Parameters

|             |              |
|-------------|--------------|
| <i>port</i> | USB-PD port. |
|-------------|--------------|

**Applicable devices:** PAG1S.

## 6.26.5.107 pd\_pasc\_set\_valley()

```
ccg_status_t pd_pasc_set_valley (
 uint8_t port,
 uint8_t valley,
 pasc_valley_cbk_t cb)
```

Update the operating valley for PWM operation. The function only triggers the transition and does not wait for completion. The callback is triggered on completion or if an error is encountered. If a valley update is called while in progress the existing request shall be aborted and callback shall not be triggered.

## Parameters

|               |                                                                          |
|---------------|--------------------------------------------------------------------------|
| <i>port</i>   | USB-PD port.                                                             |
| <i>valley</i> | New valley to move to.                                                   |
| <i>cb</i>     | Pointer to callback function to be invoked on completion of the request. |

## Returns

Status of the update request.

**Applicable devices:** PAG1S.

## 6.26.5.108 pd\_pasc\_start()

```
void pd_pasc_start (
 uint8_t port)
```

Start and initialize the secondary side regulation control (SSC). The function is invoked directly from the pd\_pasc\_init function and should not be invoked outside of the existing initialization sequence.

## Parameters

|             |              |
|-------------|--------------|
| <i>port</i> | USB-PD port. |
|-------------|--------------|

**Applicable devices:** PAG1S.

## 6.26.5.109 pd\_pasc\_valley\_algo\_enable()

```
void pd_pasc_valley_algo_enable (
 uint8_t port,
 const pasc_valley_table_t * table)
```

Initialize and start the firmware valley algorithm.

Valley algorithm is applicable only for QR operation in SSC mode PAG1S designs. The valley table provides the various valleys for optimal frequency of operation. The function does not validate the parameters and the caller is expected to provide only valid data matching the system.

## Parameters

|              |                                                           |
|--------------|-----------------------------------------------------------|
| <i>port</i>  | USB-PD port.                                              |
| <i>table</i> | The valley table to be used for performing the algorithm. |

## Returns

Status of the update request.

**Applicable devices:** PAG1S.

6.26.5.110 `pd_pasc_vbus_in_set_volt()`

```
void pd_pasc_vbus_in_set_volt (
 uint8_t port,
 uint16_t voltmV,
 PD_ADC_CB_T cbk)
```

This function initiates a transition of VBUS\_IN voltage from VSAFE\_5V to the selected voltage.

The function must be called only in un-attached state. If it needs to be aborted, then VBUS\_IN has to be first set to 5V using `pd_pasc_vbus_in_set_volt_abort()`. This implementation is available for low power mode functionality and is not expected to be called explicitly by the user.

## Parameters

|               |                                                                      |
|---------------|----------------------------------------------------------------------|
| <i>port</i>   | USB-PD port.                                                         |
| <i>voltmV</i> | Voltage to be set in mV units                                        |
| <i>cbk</i>    | The callback function to be invoked on completion of the transition. |

## Returns

None.

**Applicable devices:** PAG1S.

6.26.5.111 `pd_pasc_vbus_in_set_volt_abort()`

```
void pd_pasc_vbus_in_set_volt_abort (
 uint8_t port)
```

This function aborts and restores the VBUS\_IN transition to normal mode.

The function must be called only in un-attached state. This implementation is available for low power mode functionality and is not expected to be called explicitly by the user.

## Parameters

|             |              |
|-------------|--------------|
| <i>port</i> | USB-PD port. |
|-------------|--------------|

## Returns

None.

**Applicable devices:** PAG1S.

## 6.26.5.112 pd\_pasc\_vbus\_in\_set\_vsafe\_5V()

```
void pd_pasc_vbus_in_set_vsafe_5V (
 uint8_t port)
```

This function does blocking Vbus transition to VSAFE\_5V from lower iDAC setting to zero. It assumes that the caller has already setup the iDAC to negative value. Calling the function with a positive iDAC setting shall result in EA being loaded directly to zero and CV mode disabled. If the VBUS is higher than 5V, the VBUS\_IN safe voltage correction is expected to drop the voltage.

The function is expected to be called only during HAL start / stop functions and should not be invoked while in attached state.

## Parameters

|             |              |
|-------------|--------------|
| <i>port</i> | USB-PD port. |
|-------------|--------------|

## Returns

None.

**Applicable devices:** PAG1S.

## 6.26.5.113 pd\_phy\_abort\_bist\_cm2()

```
ccg_status_t pd_phy_abort_bist_cm2 (
 uint8_t port)
```

This function stops transmission of BIST Carrier Mode 2.

## Parameters

|             |                                                                |
|-------------|----------------------------------------------------------------|
| <i>port</i> | Port index. Caller should ensure to provide only valid values. |
|-------------|----------------------------------------------------------------|

## Returns

ccg\_status\_t

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

## 6.26.5.114 pd\_phy\_abort\_tx\_msg()

```
ccg_status_t pd_phy_abort_tx_msg (
 uint8_t port)
```

This function stops transmission of any ongoing PD message.

## Parameters

|             |                                                                |
|-------------|----------------------------------------------------------------|
| <i>port</i> | Port index. Caller should ensure to provide only valid values. |
|-------------|----------------------------------------------------------------|

## Returns

ccg\_status\_t

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

### 6.26.5.115 pd\_phy\_deepsleep()

```
bool pd_phy_deepsleep (
 uint8_t port)
```

This function configures the PD block for deepsleep entry.

#### Parameters

|             |                                                                |
|-------------|----------------------------------------------------------------|
| <i>port</i> | Port index. Caller should ensure to provide only valid values. |
|-------------|----------------------------------------------------------------|

#### Returns

Returns true if the port is not busy and has been configured to go to deepsleep, false otherwise. Also returns true if the block was not enabled.

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

### 6.26.5.116 pd\_phy\_dis\_unchunked\_tx()

```
void pd_phy_dis_unchunked_tx (
 uint8_t port)
```

This function disables transmission of unchunked extended messages.

#### Parameters

|             |                                                                |
|-------------|----------------------------------------------------------------|
| <i>port</i> | Port index. Caller should ensure to provide only valid values. |
|-------------|----------------------------------------------------------------|

#### Returns

None

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

### 6.26.5.117 pd\_phy\_en\_unchunked\_tx()

```
void pd_phy_en_unchunked_tx (
 uint8_t port)
```

This function enables transmission of unchunked extended messages.

#### Parameters

|             |                                                                |
|-------------|----------------------------------------------------------------|
| <i>port</i> | Port index. Caller should ensure to provide only valid values. |
|-------------|----------------------------------------------------------------|

#### Returns

None

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

### 6.26.5.118 pd\_phy\_get\_rx\_packet()

```
pd_packet_extd_t* pd_phy_get_rx_packet (
```

```
uint8_t port)
```

This function returns the received packet. Since the interrupt handlers uses the same buffer to receive all incoming messages, the caller of this function needs to copy the data out before a new message is received.

#### Parameters

|             |                                                                |
|-------------|----------------------------------------------------------------|
| <i>port</i> | Port index. Caller should ensure to provide only valid values. |
|-------------|----------------------------------------------------------------|

#### Returns

Pointer to the received PD packet.

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

#### 6.26.5.119 pd\_phy\_init()

```
ccg_status_t pd_phy_init (
 uint8_t port,
 pd_phy_cbk_t cbk)
```

This function initializes the PD phy registers and registers a callback (provided by the PD protocol state machine) which will be called when interrupts associated with the PD transceiver are received.

#### Parameters

|             |                                                                |
|-------------|----------------------------------------------------------------|
| <i>port</i> | Port index. Caller should ensure to provide only valid values. |
| <i>cbk</i>  | The phy event handler callback.                                |

#### Returns

ccg\_status\_t

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

#### 6.26.5.120 pd\_phy\_is\_busy()

```
bool pd_phy_is_busy (
 uint8_t port)
```

This function checks if the PD phy is busy for the specified port.

#### Parameters

|             |                                                                |
|-------------|----------------------------------------------------------------|
| <i>port</i> | Port index. Caller should ensure to provide only valid values. |
|-------------|----------------------------------------------------------------|

#### Returns

Returns true if the phy is busy, false otherwise.

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

#### 6.26.5.121 pd\_phy\_load\_msg()

```
bool pd_phy_load_msg (
```

```

uint8_t port,
sop_t sop,
uint8_t retries,
uint8_t dobj_count,
uint32_t header,
bool unchunked,
uint32_t * buf)

```

This function loads the PD message in FIFO and configures the necessary registers in preparation to sending the message out. The actual message sending has to be triggered by calling [pd\\_phy\\_send\\_msg\(\)](#).

#### Parameters

|                   |                                                                                                                          |
|-------------------|--------------------------------------------------------------------------------------------------------------------------|
| <i>port</i>       | Port index. Caller should ensure to provide only valid values.                                                           |
| <i>sop</i>        | Sop type.                                                                                                                |
| <i>retries</i>    | Number of retries.                                                                                                       |
| <i>dobj_count</i> | No of data objects(each 32 bit) in data                                                                                  |
| <i>header</i>     | PD Header in lower 16 bits and optional unchunked extended header in upper 16 bits.                                      |
| <i>unchunked</i>  | Unchunked message if true.                                                                                               |
| <i>buf</i>        | Pointer to message. Message buffer is a DWORD array that holds the data associated with data and extended data messages. |

#### Returns

Returns true if successful, false otherwise.

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

#### 6.26.5.122 pd\_phy\_refresh\_roles()

```

void pd_phy_refresh_roles (
 uint8_t port)

```

This function configures the PD phy based on the current power role, data role and contract status of the specified port. This API does not enable the receiver.

#### Parameters

|             |                                                                |
|-------------|----------------------------------------------------------------|
| <i>port</i> | Port index. Caller should ensure to provide only valid values. |
|-------------|----------------------------------------------------------------|

#### Returns

None

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

#### 6.26.5.123 pd\_phy\_reset\_rx\_tx\_sm()

```

void pd_phy_reset_rx_tx_sm (
 uint8_t port)

```

Reset the PD receive and transmit state machines.

#### Parameters

|             |                                                                |
|-------------|----------------------------------------------------------------|
| <i>port</i> | Port index. Caller should ensure to provide only valid values. |
|-------------|----------------------------------------------------------------|

**Returns**

None

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

## 6.26.5.124 pd\_phy\_send\_bist\_cm2()

```
ccg_status_t pd_phy_send_bist_cm2 (
 uint8_t port)
```

This function starts transmission of BIST Carrier Mode 2. Once this API returns successfully, the PHY continues to send the CM2 pattern until [pd\\_phy\\_abort\\_bist\\_cm2\(\)](#) is called.

**Parameters**

|             |                                                                |
|-------------|----------------------------------------------------------------|
| <i>port</i> | Port index. Caller should ensure to provide only valid values. |
|-------------|----------------------------------------------------------------|

**Returns**

ccg\_status\_t

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

## 6.26.5.125 pd\_phy\_send\_msg()

```
bool pd_phy_send_msg (
 uint8_t port)
```

This function starts the transmission of a message already loaded in FIFO.

**Parameters**

|             |                                                                |
|-------------|----------------------------------------------------------------|
| <i>port</i> | Port index. Caller should ensure to provide only valid values. |
|-------------|----------------------------------------------------------------|

**Returns**

Returns true if successful, false otherwise.

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

## 6.26.5.126 pd\_phy\_send\_reset()

```
ccg_status_t pd_phy_send_reset (
 uint8_t port,
 sop_t sop)
```

This function starts transmission of a cable reset or a hard reset as requested.

**Parameters**

|             |                                                                |
|-------------|----------------------------------------------------------------|
| <i>port</i> | Port index. Caller should ensure to provide only valid values. |
| <i>sop</i>  | SOP type indicating Cable Reset or Hard Reset.                 |

**Returns**

ccg\_status\_t

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

**6.26.5.127 pd\_phy\_wakeup()**

```
bool pd_phy_wakeup (
 void)
```

This function configures all PD blocks supported by the device after deepsleep exit.

**Returns**

Returns true if successful, false otherwise.

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

**6.26.5.128 pd\_remove\_internal\_fb\_res\_div()**

```
void pd_remove_internal_fb_res_div (
 void)
```

Removes internal feedback resistor divider.

**Returns**

None

**Applicable devices:** CCG3PA, CCG3PA2.

**6.26.5.129 pd\_reset\_edge\_det()**

```
void pd_reset_edge_det (
 uint8_t port,
 bool pgdo_type)
```

Resets the gate driver edge detector to clear any fault state.

**Parameters**

|                  |                                                                                      |
|------------------|--------------------------------------------------------------------------------------|
| <i>port</i>      | Port index.                                                                          |
| <i>pgdo_type</i> | Flag indicating the gate driver to be cleared, true for P_CTRL and false for C_CTRL. |

**Returns**

None

**Applicable devices:** CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

**6.26.5.130 pd\_sample\_pfc\_comp()**

```
bool pd_sample_pfc_comp (
 uint8_t port,
 uint32_t cur)
```



Samples the PFC comparator. The function samples against the requested current and restores the previous settings after sampling. This function is useful when the comparator is already configured for interrupt and needs to be multiplexed to sample at a different current.

#### Parameters

|             |                                            |
|-------------|--------------------------------------------|
| <i>port</i> | USB-PD port on which to sample comparator. |
| <i>cur</i>  | Operating current in 10mA units.           |

#### Returns

State of the comparator on sampling.

**Applicable devices:** CCG3PA, CCG3PA2, PAG1S.

#### 6.26.5.131 pd\_set\_pfc\_comp()

```
bool pd_set_pfc_comp (
 uint8_t port,
 uint32_t cur,
 filter_edge_detect_cfg_t edge,
 pd_cmp_cbk_t cbk)
```

Enables the PFC comparator.

#### Parameters

|             |                                               |
|-------------|-----------------------------------------------|
| <i>port</i> | USB-PD port on which to enable comparator.    |
| <i>cur</i>  | Operating current in 10mA units.              |
| <i>edge</i> | Which edge interrupt configure for.           |
| <i>cbk</i>  | Function to be called when interrupt happens. |

#### Returns

State of the comparator after configuration.

**Applicable devices:** CCG3PA, CCG3PA2, PAG1S.

#### 6.26.5.132 pd\_set\_sr\_comp()

```
bool pd_set_sr_comp (
 uint8_t port,
 uint32_t cur,
 filter_edge_detect_cfg_t edge,
 pd_cmp_cbk_t cbk)
```

Enables the SR comparator.

#### Parameters

|             |                                               |
|-------------|-----------------------------------------------|
| <i>port</i> | USB-PD port on which to enable comparator.    |
| <i>cur</i>  | Operating current in 10mA units.              |
| <i>edge</i> | Which edge interrupt configure for.           |
| <i>cbk</i>  | Function to be called when interrupt happens. |

**Returns**

State of the comparator after configuration.

**Applicable devices:** CCG3PA, CCG3PA2, PAG1S.

**6.26.5.133 pd\_srgdrv\_disable()**

```
void pd_srgdrv_disable (
 uint8_t port)
```

Disable the SR gate driver. NOTE: If the gate driver needs to be selectively disabled, then invoke this API. NOTE: If the CCG\_SRGD<sub>RV</sub>\_DISABLE\_ON\_TRANSITION macro is enabled, the gate driver has to be explicitly enabled/disabled while transition happens.

**Parameters**

|             |              |
|-------------|--------------|
| <i>port</i> | USB-PD port. |
|-------------|--------------|

**Returns**

None.

**Applicable devices:** PAG1S.

**6.26.5.134 pd\_srgdrv\_enable()**

```
void pd_srgdrv_enable (
 uint8_t port)
```

Enable SR gate driver. NOTE: If the gate driver needs to be selectively enabled, then invoke this API. NOTE: If the CCG\_SRGD<sub>RV</sub>\_DISABLE\_ON\_TRANSITION macro is enabled, the gate driver has to be explicitly enabled.

**Parameters**

|             |              |
|-------------|--------------|
| <i>port</i> | USB-PD port. |
|-------------|--------------|

**Returns**

None.

**Applicable devices:** PAG1S.

**6.26.5.135 pd\_srsense\_disable()**

```
void pd_srsense_disable (
 uint8_t port)
```

Disable the SR sensing and gate driver. NOTE: This should not be called as it is already taken care of as part of HAL handlers.

**Parameters**

|             |              |
|-------------|--------------|
| <i>port</i> | USB-PD port. |
|-------------|--------------|

**Returns**

None.

**Applicable devices:** PAG1S.

**6.26.5.136 pd\_srsense\_enable()**

```
void pd_srsense_enable (
 uint8_t port)
```

Enable SR sensing. NOTE: This should not be called as it is already taken care of as part of HAL handlers.

**Parameters**

|             |              |
|-------------|--------------|
| <i>port</i> | USB-PD port. |
|-------------|--------------|

**Returns**

None.

**Applicable devices:** PAG1S.

**6.26.5.137 pd\_stop\_pfc\_comp()**

```
void pd_stop_pfc_comp (
 uint8_t port)
```

Disables PFC comparator on the specified port.

**Parameters**

|             |              |
|-------------|--------------|
| <i>port</i> | USB-PD port. |
|-------------|--------------|

**Returns**

None

**Applicable devices:** CCG3PA, CCG3PA2, PAG1S.

**6.26.5.138 pd\_stop\_sr\_comp()**

```
void pd_stop_sr_comp (
 uint8_t port)
```

Disables SR comparator on the specified port.

**Parameters**

|             |              |
|-------------|--------------|
| <i>port</i> | USB-PD port. |
|-------------|--------------|

**Returns**

None

**Applicable devices:** CCG3PA, CCG3PA2, PAG1S.

6.26.5.139 `pd_typec_dis_dpslp_rp()`

```
void pd_typec_dis_dpslp_rp (
 uint8_t port)
```

This function disables the resistive Rp termination so that the accurate current source based Rp termination can be applied.

## Parameters

|             |                                                                |
|-------------|----------------------------------------------------------------|
| <i>port</i> | Port index. Caller should ensure to provide only valid values. |
|-------------|----------------------------------------------------------------|

## Returns

None

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

6.26.5.140 `pd_typec_dis_rd()`

```
void pd_typec_dis_rd (
 uint8_t port,
 uint8_t channel)
```

This function disables Rd termination on the specified CC line.

## Parameters

|                |                                                                |
|----------------|----------------------------------------------------------------|
| <i>port</i>    | Port index. Caller should ensure to provide only valid values. |
| <i>channel</i> | Channel index, where CC1 = 0, CC2 = 1.                         |

## Returns

None

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

6.26.5.141 `pd_typec_dis_rp()`

```
void pd_typec_dis_rp (
 uint8_t port,
 uint8_t channel)
```

This function disables Rp termination on the specified CC line.

## Parameters

|                |                                                                |
|----------------|----------------------------------------------------------------|
| <i>port</i>    | Port index. Caller should ensure to provide only valid values. |
| <i>channel</i> | Channel index, where CC1 = 0, CC2 = 1.                         |

## Returns

None.

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

## 6.26.5.142 pd\_typec\_en\_deadbat\_rd()

```
void pd_typec_en_deadbat_rd (
 uint8_t port)
```

Enable Dead Battery Rd on the specified PD port. This function is used to remove the trimmed Rd and re-enable the dead-battery Rd on the PD port prior to going through a device reset. This method is only used in dead-battery use cases, and allows device flashing while the CCG device is powered through the Type-C connection.

## Parameters

|             |                                                 |
|-------------|-------------------------------------------------|
| <i>port</i> | Port on which dead-battery Rd is to be enabled. |
|-------------|-------------------------------------------------|

## Returns

None.

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6 devices which support the dead-battery Rd (Rd-db) termination.

## 6.26.5.143 pd\_typec\_en\_dpslp\_rp()

```
void pd_typec_en_dpslp_rp (
 uint8_t port)
```

This function enables a resistive Rp termination (not accurate to match any Type-C current levels) which can be used to save power while there is no Type-C connection and the CCG device is in sleep.

## Parameters

|             |                                                                |
|-------------|----------------------------------------------------------------|
| <i>port</i> | Port index. Caller should ensure to provide only valid values. |
|-------------|----------------------------------------------------------------|

## Returns

None

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

## 6.26.5.144 pd\_typec\_en\_rd()

```
void pd_typec_en_rd (
 uint8_t port,
 uint8_t channel)
```

This function enables Rd termination on the specified CC line.

## Parameters

|                |                                                                |
|----------------|----------------------------------------------------------------|
| <i>port</i>    | Port index. Caller should ensure to provide only valid values. |
| <i>channel</i> | Channel index, where CC1 = 0, CC2 = 1.                         |

## Returns

None

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

6.26.5.145 `pd_typec_en_rp()`

```
void pd_typec_en_rp (
 uint8_t port,
 uint8_t channel,
 rp_term_t rp_val)
```

This function configures and enables Rp termination on the specified CC line.

## Parameters

|                |                                                                |
|----------------|----------------------------------------------------------------|
| <i>port</i>    | Port index. Caller should ensure to provide only valid values. |
| <i>channel</i> | Channel index, where CC1 = 0, CC2 = 1.                         |
| <i>rp_val</i>  | Rp value.                                                      |

## Returns

None.

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

6.26.5.146 `pd_typec_get_cc_status()`

```
cc_state_t pd_typec_get_cc_status (
 uint8_t port)
```

This function returns current status of both CC lines. The function identifies the current termination (Rp or Rd) applied by the CCG device and interprets the voltage on the CC line based on the current use case.

## Parameters

|             |                                                                |
|-------------|----------------------------------------------------------------|
| <i>port</i> | Port index. Caller should ensure to provide only valid values. |
|-------------|----------------------------------------------------------------|

## Returns

`cc_state_t`

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

6.26.5.147 `pd_typec_init()`

```
ccg_status_t pd_typec_init (
 uint8_t port)
```

This function initializes the Type-C registers in the PD block.

## Parameters

|             |                                                                |
|-------------|----------------------------------------------------------------|
| <i>port</i> | Port index. Caller should ensure to provide only valid values. |
|-------------|----------------------------------------------------------------|

## Returns

`ccg_status_t`

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

## 6.26.5.148 pd\_typec\_rd\_enable()

```
void pd_typec_rd_enable (
 uint8_t port)
```

This function enables the Rd termination without initializing the block completely. This is used in some Type-C fault handling use cases.

## Parameters

|             |             |
|-------------|-------------|
| <i>port</i> | Port index. |
|-------------|-------------|

## Returns

None

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

## 6.26.5.149 pd\_typec\_set\_polarity()

```
void pd_typec_set_polarity (
 uint8_t port,
 bool polarity)
```

This function sets the CC polarity for the receiver circuit.

## Parameters

|                 |                                                                                |
|-----------------|--------------------------------------------------------------------------------|
| <i>port</i>     | Port index. Caller should ensure to provide only valid values.                 |
| <i>polarity</i> | Type-C connection orientation. false when CC1 active and true when CC2 active. |

## Returns

None.

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

## 6.26.5.150 pd\_typec\_snk\_update\_trim()

```
void pd_typec_snk_update_trim (
 uint8_t port)
```

This function updates the tx trim settings when in the sink role. It must be called whenever an Rp change on the port partner side is detected. This is required to meet the CC transmitter characteristics defined by the USB-PD specification.

## Parameters

|             |             |
|-------------|-------------|
| <i>port</i> | Port index. |
|-------------|-------------|

## Returns

None

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6.

6.26.5.151 `pd_typec_start()`

```
ccg_status_t pd_typec_start (
 uint8_t port)
```

This function starts the Type C line comparators. `pdss_typec_init()` should have been called before calls to this function.

## Parameters

|             |                                                                |
|-------------|----------------------------------------------------------------|
| <i>port</i> | Port index. Caller should ensure to provide only valid values. |
|-------------|----------------------------------------------------------------|

## Returns

`ccg_status_t`

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

6.26.5.152 `pd_typec_stop()`

```
ccg_status_t pd_typec_stop (
 uint8_t port)
```

: This function stops the Type-C line comparators.

## Parameters

|             |                                                                |
|-------------|----------------------------------------------------------------|
| <i>port</i> | Port index. Caller should ensure to provide only valid values. |
|-------------|----------------------------------------------------------------|

## Returns

: `ccg_status_t`

**Applicable devices:** CCG3, CCG4, CCG5, CCG3PA, CCG3PA2, CCG5C, CCG6, PAG1S.

6.26.5.153 `pd_vconn_disable()`

```
ccg_status_t pd_vconn_disable (
 uint8_t port,
 uint8_t channel)
```

This function turns off Vconn for the specified channel.

## Parameters

|                |                                                                |
|----------------|----------------------------------------------------------------|
| <i>port</i>    | Port index. Caller should ensure to provide only valid values. |
| <i>channel</i> | Channel index, where CC1 = 0, CC2 = 1.                         |

## Returns

`ccg_status_t`

**Applicable devices:** CCG3, CCG4, CCG5, CCG5C, CCG6.



## 6.26.5.154 pd\_vconn\_enable()

```
ccg_status_t pd_vconn_enable (
 uint8_t port,
 uint8_t channel)
```

This function turns on Vconn for the specified channel.

## Parameters

|                |                                                                |
|----------------|----------------------------------------------------------------|
| <i>port</i>    | Port index. Caller should ensure to provide only valid values. |
| <i>channel</i> | Channel index, where CC1 = 0, CC2 = 1.                         |

## Returns

ccg\_status\_t

**Applicable devices:** CCG3, CCG4, CCG5, CCG5C, CCG6.

## 6.26.5.155 sbu\_switch\_configure()

```
ccg_status_t sbu_switch_configure (
 uint8_t port,
 sbu_switch_state_t sbu1_state,
 sbu_switch_state_t sbu2_state)
```

Configure the SBU switch in the CCGx device to connect the SBU1/2 to pins in the desired orientation.

## Parameters

|                   |                                                  |
|-------------------|--------------------------------------------------|
| <i>port</i>       | PD port on which SBU switch is to be configured. |
| <i>sbu1_state</i> | Desired SBU1 switch state.                       |
| <i>sbu2_state</i> | Desired SBU2 switch state.                       |

## Returns

CCG\_STAT\_SUCCESS or CCG\_STAT\_INVALID\_ARGUMENT.

**Applicable devices:** CCG3, CCG5, CCG5C, CCG6.

## 6.27 scb/i2c.h File Reference

```
#include <stdint.h>
#include <stdbool.h>
#include <config.h>
#include "timer.h"
#include "timer_id.h"
```

## Data Structures

- struct [i2c\\_scb\\_config\\_t](#)

## Macros

- `#define I2C_BLOCK_COUNT (4)`
- `#define I2C_CLEAR_INTR_MASK (0x00000000)`
- `#define I2C_CLEAR_INTR_REQUEST_REG (0xFFFFFFFFu)`
- `#define I2C_SCB_FIFO_SIZE (8u)`
- `#define I2C_SCB_RX_FIFO_SIZE (I2C_SCB_FIFO_SIZE)`
- `#define I2C_SCB_TX_FIFO_SIZE (I2C_SCB_FIFO_SIZE)`
- `#define I2C_SLAVE_ADDR_MASK_DEFAULT (0xFE)`
- `#define I2C_SLAVE_TIMER_BASE (I2C_SLAVE_SCB0_TIMER)`
- `#define I2C_SLAVE_TIMER_PERIOD (500)`

## Typedefs

- `typedef bool(* i2c_cb_fun_t) (i2c_cb_cmd_t cmd, i2c_scb_state_t i2c_state, uint16_t count)`

## Enumerations

- `enum i2c_scb_state_t {  
I2C_SCB_STATE_DISABLED = 0, I2C_SCB_STATE_INIT, I2C_SCB_STATE_IDLE, I2C_SCB_STATE_PREAMBLE,  
I2C_SCB_STATE_READ, I2C_SCB_STATE_WRITE, I2C_SCB_STATE_CLK_STRETCH, I2C_SCB_STATE_ERROR,  
I2C_SCB_NUM_STATES }`
- `enum i2c_scb_mode_t { I2C_SCB_MODE_MASTER = 0, I2C_SCB_MODE_HPI, I2C_SCB_MODE_ALP_RIDGE  
}`
- `enum i2c_scb_clock_freq_t { I2C_SCB_CLOCK_FREQ_100_KHZ = 0, I2C_SCB_CLOCK_FREQ_400_KHZ,  
I2C_SCB_CLOCK_FREQ_1_MHZ }`
- `enum i2c_cb_cmd_t {  
I2C_CB_CMD_READ = 0, I2C_CB_CMD_WRITE, I2C_CB_CMD_XFER_END, I2C_CB_CMD_TIMEOUT,  
I2C_CB_SLAVE_ADDR_MATCH }`

## Functions

- `void i2c_scb_init (uint8_t scb_index, i2c_scb_mode_t mode, i2c_scb_clock_freq_t clock_freq, uint8_t slave←  
_addr, uint8_t slave_mask, i2c_cb_fun_t cb_fun_ptr, uint8_t *scratch_buffer, uint16_t scratch_buffer_size)`
- `void i2c_scb_deinit (uint8_t scb_index)`
- `void i2c_scb_write (uint8_t scb_index, uint8_t *source_ptr, uint8_t size, uint8_t *count)`
- `void i2c_reset (uint8_t scb_index)`
- `void i2c_slave_ack_ctrl (uint8_t scb_index, bool enable)`
- `bool i2c_scb_is_idle (uint8_t scb_index)`
- `void i2c_scb_enable_wakeup (uint8_t scb_index)`
- `void i2c_timer_cb (uint8_t instance, timer_id_t id)`

### 6.27.1 Detailed Description

I2C slave driver header file.

### 6.27.2 Macro Definition Documentation

### 6.27.2.1 I2C\_BLOCK\_COUNT

```
#define I2C_BLOCK_COUNT (4)
```

Number of I2C blocks supported by the device.

The CCG3 and CCG4 device families support 4 SCB blocks which can be used for I2C functionality. CCG3PA, CCG3PA2 supports two SCB blocks. If the target device using this stack supports a different number of SCBs, this macro value will need to be updated.

### 6.27.2.2 I2C\_CLEAR\_INTR\_MASK

```
#define I2C_CLEAR_INTR_MASK (0x00000000)
```

Value to clear Interrupt mask bits.

### 6.27.2.3 I2C\_CLEAR\_INTR\_REQUEST\_REG

```
#define I2C_CLEAR_INTR_REQUEST_REG (0xFFFFFFFFu)
```

Value to clear Interrupt request bits.

### 6.27.2.4 I2C\_SCB\_FIFO\_SIZE

```
#define I2C_SCB_FIFO_SIZE (8u)
```

Size of FIFO in the SCB block for I2C transfers.

### 6.27.2.5 I2C\_SCB\_RX\_FIFO\_SIZE

```
#define I2C_SCB_RX_FIFO_SIZE (I2C_SCB_FIFO_SIZE)
```

Size of FIFO provided by the SCB block for I2C read (incoming data) transfers.

### 6.27.2.6 I2C\_SCB\_TX\_FIFO\_SIZE

```
#define I2C_SCB_TX_FIFO_SIZE (I2C_SCB_FIFO_SIZE)
```

Size of FIFO provided by the SCB block for I2C write (outgoing data) transfers.

### 6.27.2.7 I2C\_SLAVE\_ADDR\_MASK\_DEFAULT

```
#define I2C_SLAVE_ADDR_MASK_DEFAULT (0xFE)
```

I2C Slave address mask to be applied on received preamble.

### 6.27.2.8 I2C\_SLAVE\_TIMER\_BASE

```
#define I2C_SLAVE_TIMER_BASE (I2C_SLAVE_SCB0_TIMER)
```

Base ID of timers reserved for I2C transfer timeout implementation.

### 6.27.2.9 I2C\_SLAVE\_TIMER\_PERIOD

```
#define I2C_SLAVE_TIMER_PERIOD (500)
```

Timeout period for I2C transfers in milliseconds. The I2C block will be reset if any transaction does not complete within this time period.

### 6.27.3 Enumeration Type Documentation

#### 6.27.3.1 i2c\_cb\_cmd\_t

enum `i2c_cb_cmd_t`

Type of I2C operation being notified through a callback function.

##### Enumerator

|                                      |                                                           |
|--------------------------------------|-----------------------------------------------------------|
| <code>I2C_CB_CMD_READ</code>         | Read command from master.                                 |
| <code>I2C_CB_CMD_WRITE</code>        | Write command from master.                                |
| <code>I2C_CB_CMD_XFER_END</code>     | End of read transfer: STOP condition signalled by master. |
| <code>I2C_CB_CMD_TIMEOUT</code>      | Timeout on I2C operation.                                 |
| <code>I2C_CB_SLAVE_ADDR_MATCH</code> | I2C slave address match detected.                         |

#### 6.27.3.2 i2c\_scb\_clock\_freq\_t

enum `i2c_scb_clock_freq_t`

List of possible I2C bus bit rates.

##### Enumerator

|                                         |                    |
|-----------------------------------------|--------------------|
| <code>I2C_SCB_CLOCK_FREQ_100_KHZ</code> | 100 KHz operation. |
| <code>I2C_SCB_CLOCK_FREQ_400_KHZ</code> | 400 KHz operation. |
| <code>I2C_SCB_CLOCK_FREQ_1_MHZ</code>   | 1 MHz operation.   |

#### 6.27.3.3 i2c\_scb\_mode\_t

enum `i2c_scb_mode_t`

List of possible I2C block operating modes.

This type lists the possible I2C block operating modes. Multiple slave modes are defined in case there is any mode specific handling required for any of them.

Note: Master mode is currently not supported by the driver. Note: The HPI and ALP\_RIDGE mode handling is equivalent.

##### Enumerator

|                                     |                                                                    |
|-------------------------------------|--------------------------------------------------------------------|
| <code>I2C_SCB_MODE_MASTER</code>    | Master mode: Can be used for mux control. Not supported as of now. |
| <code>I2C_SCB_MODE_HPI</code>       | Slave mode: Used for HPI interface.                                |
| <code>I2C_SCB_MODE_ALP_RIDGE</code> | Slave mode: Used for Alpine Ridge interface.                       |

## 6.27.3.4 i2c\_scb\_state\_t

```
enum i2c_scb_state_t
```

List of possible I2C block states.

The I2C driver implements a state machine which ensures that read/write transfers are responded to correctly without any corruption. This data type lists the possible states in this slave mode state machine.

## Enumerator

|                           |                                                           |
|---------------------------|-----------------------------------------------------------|
| I2C_SCB_STATE_DISABLED    | I2C interface is disabled.                                |
| I2C_SCB_STATE_INIT        | Interface initialized and waiting to be enabled.          |
| I2C_SCB_STATE_IDLE        | Interface ready and waiting for preamble from the master. |
| I2C_SCB_STATE_PREAMBLE    | Preamble phase is in progress.                            |
| I2C_SCB_STATE_READ        | I2C read operation is in progress.                        |
| I2C_SCB_STATE_WRITE       | I2C write operation is in progress.                       |
| I2C_SCB_STATE_CLK_STRETCH | Drive is stretching I2C clock to delay operation.         |
| I2C_SCB_STATE_ERROR       | Error state: transaction error detected.                  |
| I2C_SCB_NUM_STATES        | Last state ID: not used.                                  |

## 6.27.4 Function Documentation

## 6.27.4.1 i2c\_reset()

```
void i2c_reset (
 uint8_t scb_index)
```

Reset the I2C block specified.

This function resets the I2C block in response to an error or explicit request from protocol layer.

## Parameters

|                  |                     |
|------------------|---------------------|
| <i>scb_index</i> | SCB ID to be reset. |
|------------------|---------------------|

## Returns

None

## 6.27.4.2 i2c\_scb\_deinit()

```
void i2c_scb_deinit (
 uint8_t scb_index)
```

De-initialize a previously initialized SCB block.

## Parameters

|                  |                           |
|------------------|---------------------------|
| <i>scb_index</i> | SCB index to be disabled. |
|------------------|---------------------------|

**Returns**

None

**6.27.4.3 i2c\_scb\_enable\_wakeup()**

```
void i2c_scb_enable_wakeup (
 uint8_t scb_index)
```

Enable deep-sleep wakeup due to address match on the specified SCB block.

**Parameters**

|                  |                                                       |
|------------------|-------------------------------------------------------|
| <i>scb_index</i> | SCB ID to be configured as deep-sleep wakeup trigger. |
|------------------|-------------------------------------------------------|

**Returns**

None

**6.27.4.4 i2c\_scb\_init()**

```
void i2c_scb_init (
 uint8_t scb_index,
 i2c_scb_mode_t mode,
 i2c_scb_clock_freq_t clock_freq,
 uint8_t slave_addr,
 uint8_t slave_mask,
 i2c_cb_fun_t cb_fun_ptr,
 uint8_t * scratch_buffer,
 uint16_t scratch_buffer_size)
```

Configure one of the I2C blocks as required.

This API is used to enable and configure one of the I2C blocks for driver operation. Only I2C slave operation is currently supported by the driver as of now.

The I2C driver is agnostic of the actual data transfer protocol. It reads all data written by the master into a receive buffer provided by the protocol layer. A callback function is used to notify the protocol layer when the write is complete. The receive buffer provided should be big enough to hold the maximum amount of data that the master may provide in a write operation. If the write contains more data than the buffer can hold, the I2C driver will NAK the transaction.

Read requests from the I2C master are automatically delayed by clock stretching. A callback function is used to notify the protocol layer that the master is waiting for data. The `i2c_scb_write` function can be used by the protocol layer to write data into the transmit FIFO in response to the read request.

All I2C driver events are generated from interrupt context, and are expected to be handled with care. The protocol layer should defer any long operations to a non-interrupt context.

**Parameters**

|                   |                                                       |
|-------------------|-------------------------------------------------------|
| <i>scb_index</i>  | SCB index being configured for I2C operation.         |
| <i>mode</i>       | Desired mode of operation.                            |
| <i>clock_freq</i> | Desired I2C clock frequency.                          |
| <i>slave_addr</i> | Device address to be used in case of slave operation. |

## Parameters

|                            |                                                        |
|----------------------------|--------------------------------------------------------|
| <i>slave_mask</i>          | Mask to be applied on for slave address matching.      |
| <i>cb_fun_ptr</i>          | Callback function to be called for event notification. |
| <i>scratch_buffer</i>      | Receive buffer used to hold written by master.         |
| <i>scratch_buffer_size</i> | Size of the receive buffer in bytes.                   |

## Returns

None

## 6.27.4.5 i2c\_scb\_is\_idle()

```
bool i2c_scb_is_idle (
 uint8_t scb_index)
```

Check whether the I2C block is idle.

This function checks whether the specified I2C block is idle. This check should be performed before the device enters deep sleep. Deep sleep entry should be avoided if this function returns false.

## Parameters

|                  |                       |
|------------------|-----------------------|
| <i>scb_index</i> | SCB ID to be checked. |
|------------------|-----------------------|

## Returns

true if the I2C block is idle, false otherwise.

## 6.27.4.6 i2c\_scb\_write()

```
void i2c_scb_write (
 uint8_t scb_index,
 uint8_t * source_ptr,
 uint8_t size,
 uint8_t * count)
```

Write data into the transmit FIFO associated with the I2C block.

This function is used by the protocol layer to write data into the I2C transmit FIFO in response to a master read operation.

## Parameters

|                   |                                                                      |
|-------------------|----------------------------------------------------------------------|
| <i>scb_index</i>  | SCB ID to do the I2C transfer through.                               |
| <i>source_ptr</i> | Pointer to buffer containing the data to be written.                 |
| <i>size</i>       | Size of the data buffer. Maximum amount of data that may be written. |
| <i>count</i>      | Return parameter through which the actual write size is returned.    |

**Returns**

None

**6.27.4.7 i2c\_slave\_ack\_ctrl()**

```
void i2c_slave_ack_ctrl (
 uint8_t scb_index,
 bool enable)
```

Enable/Disable the I2C slave acknowledgement.

This function enables/disables the slave address ACK from the I2C block. The protocol layer can disable the address ACK to hold off data transfers when it is not ready to respond to the master.

**Parameters**

|                  |                                                                      |
|------------------|----------------------------------------------------------------------|
| <i>scb_index</i> | SCB ID to configure.                                                 |
| <i>enable</i>    | Whether to enable or disable the auto slave address acknowledgement. |

**Returns**

None

**6.27.4.8 i2c\_timer\_cb()**

```
void i2c_timer_cb (
 uint8_t instance,
 timer_id_t id)
```

Timer callback that indicates I2C transaction that has timed out.

**Parameters**

|                 |                                                           |
|-----------------|-----------------------------------------------------------|
| <i>instance</i> | The Timer instance associated with this timer. Will be 0. |
| <i>id</i>       | The timer id used to time I2C transactions.               |

**Returns**

None

**6.28 system/boot.h File Reference**

```
#include "stdint.h"
#include "stdbool.h"
#include "config.h"
#include "status.h"
#include "system.h"
```



## Data Structures

- union [fw\\_img\\_status\\_t](#)
- struct [fw\\_img\\_status\\_t::fw\\_mode\\_reason\\_t](#)
- struct [sys\\_fw\\_metadata\\_t](#)

## Macros

- #define [CY\\_PD\\_IMG1\\_FW\\_STATUS\\_BIT\\_MASK](#) (0x08)
- #define [CCG\\_BOOT\\_MODE\\_RQT\\_SIG](#) (0x424C)
- #define [CCG\\_FW1\\_BOOT\\_RQT\\_SIG](#) (0x4231)
- #define [CCG\\_FW2\\_BOOT\\_RQT\\_SIG](#) (0x4232)
- #define [CCG\\_FW\\_METADATA\\_BOOTSEQ\\_OFFSET](#) (0x1C)
- #define [CCG\\_BL\\_WAIT\\_DEFAULT](#) (50)
- #define [CCG\\_BL\\_WAIT\\_MINIMUM](#) (20)
- #define [CCG\\_BL\\_WAIT\\_MAXIMUM](#) (1000)
- #define [CCG\\_FWMETA\\_APPID\\_WAIT\\_DEF](#) (0xFFFFu)
- #define [CCG\\_FWMETA\\_APPID\\_WAIT\\_0](#) (0x4359)
- #define [CONFIGTABLE\\_SIGNATURE](#) (0x4359)
- #define [CONFIGTABLE\\_SIZE\\_OFFSET](#) (6)
- #define [CONFIGTABLE\\_CHECKSUM\\_OFFSET](#) (8)
- #define [CONFIGTABLE\\_CHECKSUM\\_START](#) (10)

## Functions

- [ccg\\_status\\_t boot\\_validate\\_configtable](#) (uint8\_t \*table\_p)
- [ccg\\_status\\_t boot\\_validate\\_fw](#) (sys\_fw\_metadata\_t \*fw\_metadata)
- [ccg\\_status\\_t boot\\_handle\\_validate\\_fw\\_cmd](#) (sys\_fw\_mode\_t fw\_mode)
- [uint16\\_t boot\\_get\\_wait\\_time](#) (void)
- [bool boot\\_start](#) (void)
- [void boot\\_check\\_for\\_valid\\_fw](#) (void)
- [fw\\_img\\_status\\_t get\\_boot\\_mode\\_reason](#) (void)
- [void boot\\_jump\\_to\\_fw](#) (void)
- [uint32\\_t boot\\_get\\_boot\\_seq](#) (uint8\_t fwid)
- [void boot\\_update\\_fw\\_status](#) (void)

## Variables

- [sys\\_fw\\_metadata\\_t \\* gl\\_img1\\_fw\\_metadata](#)
- [sys\\_fw\\_metadata\\_t \\* gl\\_img2\\_fw\\_metadata](#)
- [sys\\_fw\\_metadata\\_t \\* gl\\_img1\\_fw\\_pseudo\\_metadata](#)
- [sys\\_fw\\_metadata\\_t \\* gl\\_img2\\_fw\\_pseudo\\_metadata](#)
- [fw\\_img\\_status\\_t gl\\_img\\_status](#)

### 6.28.1 Detailed Description

Bootloader support header file.

### 6.28.2 Macro Definition Documentation

### 6.28.2.1 CCG\_BL\_WAIT\_DEFAULT

```
#define CCG_BL_WAIT_DEFAULT (50)
```

Default boot-wait window for CCGx boot-loader: 50 ms

### 6.28.2.2 CCG\_BL\_WAIT\_MAXIMUM

```
#define CCG_BL_WAIT_MAXIMUM (1000)
```

Maximum boot-wait window duration supported: 1000 ms

### 6.28.2.3 CCG\_BL\_WAIT\_MINIMUM

```
#define CCG_BL_WAIT_MINIMUM (20)
```

Minimum boot-wait window duration supported: 20 ms

### 6.28.2.4 CCG\_BOOT\_MODE\_RQT\_SIG

```
#define CCG_BOOT_MODE_RQT_SIG (0x424C)
```

Signature used for firmware to indicate boot mode request.

### 6.28.2.5 CCG\_FW1\_BOOT\_RQT\_SIG

```
#define CCG_FW1_BOOT_RQT_SIG (0x4231)
```

Signature used to indicate boot FW1 request.

### 6.28.2.6 CCG\_FW2\_BOOT\_RQT\_SIG

```
#define CCG_FW2_BOOT_RQT_SIG (0x4232)
```

Signature used to indicate boot FW2 request.

### 6.28.2.7 CCG\_FW\_METADATA\_BOOTSEQ\_OFFSET

```
#define CCG_FW_METADATA_BOOTSEQ_OFFSET (0x1C)
```

Firmware boot sequence number offset.

### 6.28.2.8 CCG\_FWMETA\_APPID\_WAIT\_0

```
#define CCG_FWMETA_APPID_WAIT_0 (0x4359)
```

FW metadata application ID value requesting a zero boot-wait window.

### 6.28.2.9 CCG\_FWMETA\_APPID\_WAIT\_DEF

```
#define CCG_FWMETA_APPID_WAIT_DEF (0xFFFFu)
```

FW metadata application ID value requesting default boot-wait window.

#### 6.28.2.10 CONFIGTABLE\_CHECKSUM\_OFFSET

```
#define CONFIGTABLE_CHECKSUM_OFFSET (8)
```

Offset to checksum field in config table.

#### 6.28.2.11 CONFIGTABLE\_CHECKSUM\_START

```
#define CONFIGTABLE_CHECKSUM_START (10)
```

Offset at which table checksum calculation starts.

#### 6.28.2.12 CONFIGTABLE\_SIGNATURE

```
#define CONFIGTABLE_SIGNATURE (0x4359)
```

Signature used to validate config table content: "CY"

#### 6.28.2.13 CONFIGTABLE\_SIZE\_OFFSET

```
#define CONFIGTABLE_SIZE_OFFSET (6)
```

Offset to table size field in config table.

#### 6.28.2.14 CY\_PD\_IMG1\_FW\_STATUS\_BIT\_MASK

```
#define CY_PD_IMG1_FW_STATUS_BIT_MASK (0x08)
```

Mask for Image-1 FW status bit in Boot mode reason byte.

### 6.28.3 Function Documentation

#### 6.28.3.1 boot\_check\_for\_valid\_fw()

```
void boot_check_for_valid_fw (
 void)
```

Check for the presence of alternate firmware waiting to be validated.

This function checks whether the CCGx device flash contains an alternate firmware binary which is waiting to be validated. This can happen if a firmware update happened during the last power up of the device, and the binary is yet to be validated and made active. The active firmware will make use of the pseudo metadata in flash to identify the alternate firmware, validate it and activate it by updating the actual firmware metadata.

Please refer to the CCGx boot sequence description in the firmware guide for a more detailed description of the boot procedure.

#### Returns

NONE

### 6.28.3.2 boot\_get\_boot\_seq()

```
uint32_t boot_get_boot_seq (
 uint8_t fwid)
```

Get the boot sequence number value for the specified firmware image.

A boot sequence number field stored in the firmware metadata is used by the CCGx boot-loader to identify the firmware binary to be loaded. This function retrieves the sequence number associated with the specified firmware binary.

#### Parameters

|             |                                                                                |
|-------------|--------------------------------------------------------------------------------|
| <i>fwid</i> | Firmware id whose sequence number is to be retrieved. 1 for FW1 and 2 for FW2. |
|-------------|--------------------------------------------------------------------------------|

#### Returns

Boot sequence number value if the firmware is valid, 0 otherwise.

### 6.28.3.3 boot\_get\_wait\_time()

```
uint16_t boot_get_wait_time (
 void)
```

Returns the boot-wait delay configured for the application.

This function identifies the boot-wait delay required by checking the firmware metadata.

#### Returns

Boot-wait delay in milliseconds.

### 6.28.3.4 boot\_handle\_validate\_fw\_cmd()

```
ccg_status_t boot_handle_validate_fw_cmd (
 sys_fw_mode_t fw_mode)
```

Handles the VALIDATE\_FW command from HPI or UVDM.

This API handles the VALIDATE\_FW command received through the HPI or UVDM interfaces.

#### Parameters

|                |                                              |
|----------------|----------------------------------------------|
| <i>fw_mode</i> | Firmware binary id: 1 for FW1 and 2 for FW2. |
|----------------|----------------------------------------------|

#### Returns

Status code indicating the validity of the firmware.

### 6.28.3.5 boot\_jump\_to\_fw()

```
void boot_jump_to_fw (
```

```
void)
```

Transfer control to the firmware binary identified by `boot_start`.

This function is only used by the CCGx boot-loader. This transfers control to the firmware binary selected as the boot target by the `boot_start` function. This is expected to be called after the boot-wait window has elapsed.

#### Returns

None

#### 6.28.3.6 boot\_start()

```
bool boot_start (
 void)
```

Identify the firmware binary to be loaded.

This function is only used in the CCGx boot-loader, and implements the main start-up logic of the boot-loader. The function validates the two firmware binaries in device flash, and identifies the binary to be loaded. If neither binary is valid, the function returns false notifying the caller to continue in boot-loader mode.

#### Returns

true if firmware load is allowed, false otherwise.

#### 6.28.3.7 boot\_update\_fw\_status()

```
void boot_update_fw_status (
 void)
```

Function to validate firmware images and update image status.

This function is used to validate the firmware images in the CCG device flash and update their status in the image status / boot-mode reason field.

#### Returns

None

#### 6.28.3.8 boot\_validate\_configtable()

```
ccg_status_t boot_validate_configtable (
 uint8_t * table_p)
```

Validate the configuration table specified.

Each copy of CCGx firmware on the device flash contains an embedded configuration table that defines the runtime behaviour of the CCGx device. This function checks whether the configuration table located at the specified location is valid (has valid signature and checksum).

#### Parameters

|                      |                                                     |
|----------------------|-----------------------------------------------------|
| <i>table</i> ↔<br>_p | Pointer to the configuration table to be validated. |
|----------------------|-----------------------------------------------------|

**Returns**

CCG\_STAT\_SUCCESS if the table is valid, CCG\_STAT\_FAILURE otherwise.

**6.28.3.9 boot\_validate\_fw()**

```
ccg_status_t boot_validate_fw (
 sys_fw_metadata_t * fw_metadata)
```

Validate the firmware image associated with the given metadata.

This function validates the firmware binary associated with the metadata specified in the `fw_metadata` parameter. The validity check includes checks for signature, location, size and checksum. This function internally performs validation of the embedded configuration table using the `boot_validate_configtable` function.

**Parameters**

|                          |                                                                |
|--------------------------|----------------------------------------------------------------|
| <code>fw_metadata</code> | Pointer to metadata table of the FW which has to be validated. |
|--------------------------|----------------------------------------------------------------|

**Returns**

CCG\_STAT\_SUCCESS if the firmware is valid, CCG\_STAT\_FAILURE otherwise.

**6.28.3.10 get\_boot\_mode\_reason()**

```
fw_img_status_t get_boot_mode_reason (
 void)
```

Returns a bitmap containing the reason for boot mode.

This function returns the bitmap value that is to be stored in the `BOOT_MODE_REASON` HPI register, which identifies the validity of the two firmware binaries. The validation of the firmware is expected to have been completed earlier through the `boot_start` function. This function only retrieves the status stored during the validation procedure.

**See also**

[fw\\_img\\_status\\_t](#)

**Returns**

Boot mode reason bitmap.

**6.28.4 Variable Documentation****6.28.4.1 gl\_img1\_fw\_metadata**

```
sys_fw_metadata_t* gl_img1_fw_metadata
```

Pointer to metadata associated with the Image-1 FW binary.

#### 6.28.4.2 gl\_img1\_fw\_pseudo\_metadata

`sys_fw_metadata_t*` gl\_img1\_fw\_pseudo\_metadata

Pointer to pseudo metadata associated with the Image-1 FW binary.

#### 6.28.4.3 gl\_img2\_fw\_metadata

`sys_fw_metadata_t*` gl\_img2\_fw\_metadata

Pointer to metadata associated with the Image-2 FW binary.

Pointer to metadata associated with the Image-2 FW binary.

#### 6.28.4.4 gl\_img2\_fw\_pseudo\_metadata

`sys_fw_metadata_t*` gl\_img2\_fw\_pseudo\_metadata

Pointer to pseudo metadata associated with the Image-2 FW binary.

#### 6.28.4.5 gl\_img\_status

`fw_img_status_t` gl\_img\_status

Current firmware image status.

## 6.29 system/ccgx\_host\_sdk\_desc.h File Reference

### 6.29.1 Detailed Description

CCGx Host SDK API Reference Guide Introduction.

## 6.30 system/ccgx\_version.h File Reference

### Macros

- #define [FW\\_MAJOR\\_VERSION](#) (3)
- #define [FW\\_MINOR\\_VERSION](#) (4)
- #define [FW\\_PATCH\\_VERSION](#) (0)
- #define [FW\\_BUILD\\_NUMBER](#) (2557)
- #define [FW\\_BASE\\_VERSION](#)

### 6.30.1 Detailed Description

This file defines the base firmware stack version for CCGx.

### 6.30.2 Macro Definition Documentation

### 6.30.2.1 FW\_BASE\_VERSION

```
#define FW_BASE_VERSION
```

#### Value:

```
((FW_MAJOR_VERSION << 28) | (FW_MINOR_VERSION << 24) | \
 (FW_PATCH_VERSION << 16) | (FW_BUILD_NUMBER))
```

Composite firmware stack version value.

Composite firmware stack version value. This is a 4 byte value with the following format: Bytes 1-0: Build number  
Byte 2: Patch version Byte 3 (Bits 0:3): Minor Version Byte 3 (Bits 4:7): Major Version

## 6.31 system/flash.h File Reference

```
#include "stdint.h"
#include "stdbool.h"
#include "config.h"
#include "status.h"
```

### Macros

- #define [SROM\\_API\\_PARAM\\_SIZE](#) (8u)

### Typedefs

- typedef void(\* [flash\\_cbk\\_t](#)) ([flash\\_write\\_status\\_t](#) status)

### Enumerations

- enum [flash\\_app\\_priority\\_t](#) { [FLASH\\_APP\\_PRIORITY\\_DEFAULT](#) = 0, [FLASH\\_APP\\_PRIORITY\\_IMAGE\\_1](#), [FLASH\\_APP\\_PRIORITY\\_IMAGE\\_2](#) }
- enum [flash\\_interface\\_t](#) { [FLASH\\_IF\\_HPI](#) = 0, [FLASH\\_IF\\_UVDM](#), [FLASH\\_IF\\_USB\\_HID](#), [FLASH\\_IF\\_IECS\\_UART](#) }
- enum [flash\\_write\\_status\\_t](#) { [FLASH\\_WRITE\\_COMPLETE](#), [FLASH\\_WRITE\\_ABORTED](#), [FLASH\\_WRITE\\_COMPLETE\\_AND\\_A](#), [FLASH\\_WRITE\\_IN\\_PROGRESS](#) }

### Functions

- void [flash\\_enter\\_mode](#) (bool is\_enable, [flash\\_interface\\_t](#) mode, bool data\_in\_place)
- bool [flash\\_access\\_get\\_status](#) (uint8\_t modes)
- void [flash\\_set\\_access\\_limits](#) (uint16\_t start\_row, uint16\_t last\_row, uint16\_t md\_row, uint16\_t bl\_last\_row)
- void [flash\\_get\\_access\\_limits](#) (uint16\_t \*access\_limit\_0, uint16\_t \*access\_limit\_1, uint16\_t \*access\_limit\_2, uint16\_t \*access\_limit\_3)
- [ccg\\_status\\_t](#) [flash\\_row\\_clear](#) (uint16\_t row\_num)
- [ccg\\_status\\_t](#) [flash\\_row\\_write](#) (uint16\_t row\_num, uint8\_t \*data, [flash\\_cbk\\_t](#) cbk)
- [ccg\\_status\\_t](#) [sflash\\_row\\_write](#) (uint16\_t row\_num, uint8\_t \*data)
- [ccg\\_status\\_t](#) [sflash\\_row\\_read](#) (uint16\_t row\_num, uint8\_t \*data)
- [ccg\\_status\\_t](#) [flash\\_row\\_read](#) (uint16\_t row\_num, uint8\_t \*data)
- [ccg\\_status\\_t](#) [flash\\_set\\_app\\_priority](#) ([flash\\_app\\_priority\\_t](#) app\_priority)



### 6.31.1 Detailed Description

Flash command handler header file.

### 6.31.2 Macro Definition Documentation

#### 6.31.2.1 SROM\_API\_PARAM\_SIZE

```
#define SROM_API_PARAM_SIZE (8u)
```

Size of flash write SROM API parameters in bytes.

### 6.31.3 Typedef Documentation

#### 6.31.3.1 flash\_cbk\_t

```
flash_cbk_t
```

Non-blocking flash write row callback function type.

The CCG3 device supports non-blocking flash update operations, so that the firmware can perform other operations while a flash row write is in progress. The completion of the flash row write is notified through a callback function. This type represents the function type which can be registered as a callback for notification of non-blocking flash operations.

### 6.31.4 Enumeration Type Documentation

#### 6.31.4.1 flash\_app\_priority\_t

```
enum flash_app_priority_t
```

Enumeration of app priority values.

App priority is a debug feature that allows the CCG bootloader to prioritize one copy of the firmware over the other. The default behaviour is for both firmware binaries to have the same priority so that the more recently updated firmware binary gets loaded. The priority scheme can be updated to allow FW1 or FW2 to always be loaded for debugging purposes.

Enumerator

|                                 |                                      |
|---------------------------------|--------------------------------------|
| FLASH_APP_PRIORITY_DEFAULT      | Default. Latest image gets priority. |
| FLASH_APP_PRIORITY_IMAGE↔<br>_1 | Image-1 gets priority over image-2.  |
| FLASH_APP_PRIORITY_IMAGE↔<br>_2 | Image-2 gets priority over image-1.  |

### 6.31.4.2 flash\_interface\_t

enum `flash_interface_t`

List of supported flash update interfaces.

CCG supports multiple flash update interfaces such as HPI, UVDM, USB etc. depending on the application. This enumerated type lists the supported flash update interfaces.

#### Enumerator

|                    |                                     |
|--------------------|-------------------------------------|
| FLASH_IF_HPI       | HPI based flash update interface.   |
| FLASH_IF_UVDM      | UVDM based flash update interface.  |
| FLASH_IF_USB_HID   | USB flash update interface.         |
| FLASH_IF_IECS_UART | IECS (UART) flash update interface. |

### 6.31.4.3 flash\_write\_status\_t

enum `flash_write_status_t`

List of possible status codes for non blocking flash write operation.

#### Enumerator

|                                  |                                              |
|----------------------------------|----------------------------------------------|
| FLASH_WRITE_COMPLETE             | Flash Write successfully completed.          |
| FLASH_WRITE_ABORTED              | Flash Write aborted.                         |
| FLASH_WRITE_COMPLETE_AND_ABORTED | Flash Write completed with an abort request. |
| FLASH_WRITE_IN_PROGRESS          | Flash Write is active.                       |

## 6.31.5 Function Documentation

### 6.31.5.1 flash\_access\_get\_status()

```
bool flash_access_get_status (
 uint8_t modes)
```

Check whether flashing mode has been entered.

This function checks whether flashing mode has currently been entered by a different interface.

#### Parameters

|              |                                                      |
|--------------|------------------------------------------------------|
| <i>modes</i> | Bitmap containing flashing interfaces to be checked. |
|--------------|------------------------------------------------------|

#### Returns

Returns true if flashing mode has been entered using any of the interfaces listed in modes, false otherwise.

## 6.31.5.2 flash\_enter\_mode()

```
void flash_enter_mode (
 bool is_enable,
 flash_interface_t mode,
 bool data_in_place)
```

Handle ENTER\_FLASHING\_MODE Command.

This function notifies the CCG stack that flash read/write is being enabled by the application. By default, CCG firmware disallows all flash read/write operations. Flash access is only allowed after flashing mode has been explicitly enabled through user command.

Note: FW update interface is allowed to Enable Flashing mode only once in one session. This is to ensure that multiple flashing interfaces are not active simultaneously. Each FW update interface is expected to take care of this. Once Flash access is complete, this API should be used to exit FW Update interface.

## Parameters

|                      |                                                                                                  |
|----------------------|--------------------------------------------------------------------------------------------------|
| <i>is_enable</i>     | Enable/Disable Flashing Mode                                                                     |
| <i>mode</i>          | Flash update interface to be used.                                                               |
| <i>data_in_place</i> | Specifies whether the flash write data buffer can be used in place as SROM API parameter buffer. |

## Returns

None

## 6.31.5.3 flash\_get\_access\_limits()

```
void flash_get_access_limits (
 uint16_t * access_limit_0,
 uint16_t * access_limit_1,
 uint16_t * access_limit_2,
 uint16_t * access_limit_3)
```

Extract current flash access limits.

## Parameters

|                       |                                      |
|-----------------------|--------------------------------------|
| <i>access_limit_0</i> | Start row num for flash write access |
| <i>access_limit_1</i> | Last row num for flash write access  |
| <i>access_limit_2</i> | Metadata row num                     |
| <i>access_limit_3</i> | Start row num for read access        |

## Returns

None

#### 6.31.5.4 flash\_row\_clear()

```
ccg_status_t flash_row_clear (
 uint16_t row_num)
```

Erase the contents of the specified flash row.

This API erases the contents of the specified flash row by filling it with zeroes. Please note that this API will only work if flashing mode is enabled and the selected row is within the range of write enabled rows.

##### Parameters

|                |                          |
|----------------|--------------------------|
| <i>row_num</i> | Row number to be erased. |
|----------------|--------------------------|

##### Returns

Status of row erase operation.

#### 6.31.5.5 flash\_row\_read()

```
ccg_status_t flash_row_read (
 uint16_t row_num,
 uint8_t * data)
```

Read the contents of the specified flash row.

This API handles the flash read row operation. The contents of the flash row are copied into the specified data buffer, if flashing mode has been entered and the *row\_num* is part of the readable range of memory.

##### Parameters

|                |                                            |
|----------------|--------------------------------------------|
| <i>row_num</i> | Flash row to be read.                      |
| <i>data</i>    | Buffer to read the flash row content into. |

##### Returns

Status of the flash read. CCG\_STAT\_SUCCESS or appropriate error code.

#### 6.31.5.6 flash\_row\_write()

```
ccg_status_t flash_row_write (
 uint16_t row_num,
 uint8_t * data,
 flash_cbk_t cbk)
```

Write the given data to the specified flash row.

This API handles the flash write row operation. The contents from the data buffer is written to the *row\_num* flash row. The access rules for the flash row as the same as for the *flash\_row\_clear* API.

For non-blocking write row operation, the API returns as soon as the row update is started. The stack takes care of executing all of the steps across multiple resume interrupts; and the callback is called at the end of the process.

## Parameters

|                |                                                                        |
|----------------|------------------------------------------------------------------------|
| <i>row_num</i> | Flash row to be updated.                                               |
| <i>data</i>    | Buffer containing data to be written to the flash row.                 |
| <i>cbk</i>     | Callback function to be called at the end of non-blocking flash write. |

## Returns

Status of the flash write. CCG\_STAT\_SUCCESS or appropriate error code.

## 6.31.5.7 flash\_set\_access\_limits()

```
void flash_set_access_limits (
 uint16_t start_row,
 uint16_t last_row,
 uint16_t md_row,
 uint16_t bl_last_row)
```

Set limits to the flash rows that can be accessed.

The CCG stack has been designed to support fail-safe firmware upgrades using a pair of firmware binaries that can mutually update each other. This scheme can only be robust if the currently active firmware binary can effectively protect itself by not allowing access to any of its own flash rows.

This function is used to specify the list of flash rows that can safely be accessed by the currently active firmware binary. This function should only be used with parameters derived based on the binary locations identified from firmware metadata. Incorrect usage of this API can cause the device to stop responding during a flash update operation.

This API must be invoked as part of initialization before the [flash\\_row\\_write\(\)](#) and [flash\\_row\\_read\(\)](#) functions can be called. By default, no flash row can be read or written to.

## Parameters

|                    |                                                     |
|--------------------|-----------------------------------------------------|
| <i>start_row</i>   | The lowest row number that can be written.          |
| <i>last_row</i>    | The highest row number that can be written.         |
| <i>md_row</i>      | Row containing metadata for the alternate firmware. |
| <i>bl_last_row</i> | Last bootloader row. Rows above this can be read.   |

## Returns

None

## 6.31.5.8 flash\_set\_app\_priority()

```
ccg_status_t flash_set_app_priority (
 flash_app_priority_t app_priority)
```

Updates the app boot priority flag.

This function is used to set the app priority field to override the default FW selection algorithm.

**Parameters**

|                     |                                |
|---------------------|--------------------------------|
| <i>app_priority</i> | Desired boot priority setting. |
|---------------------|--------------------------------|

**Returns**

Status code of the priority update.

**6.31.5.9 sflash\_row\_read()**

```
ccg_status_t sflash_row_read (
 uint16_t row_num,
 uint8_t * data)
```

Read the contents of the specified User SFLASH row.

**Parameters**

|                |                                             |
|----------------|---------------------------------------------|
| <i>row_num</i> | Flash row to be read.                       |
| <i>data</i>    | Buffer to read the SFLASH row content into. |

**Returns**

Status of the flash read. CCG\_STAT\_SUCCESS or appropriate error code.

**6.31.5.10 sflash\_row\_write()**

```
ccg_status_t sflash_row_write (
 uint16_t row_num,
 uint8_t * data)
```

Write the given data to the specified User SFLASH row.

This API handles writes to the user region in SFLASH of the CCG device. The function ensures that SFLASH rows containing Cypress proprietary configuration and test data is not cleared.

**Parameters**

|                |                                                         |
|----------------|---------------------------------------------------------|
| <i>row_num</i> | Flash row to be updated.                                |
| <i>data</i>    | Buffer containing data to be written to the SFLASH row. |

**Returns**

Status of the flash write. CCG\_STAT\_SUCCESS or appropriate error code.

**6.32 system/gpio.h File Reference**

```
#include "config.h"
#include "stdint.h"
#include "stdbool.h"
```

```
#include "status.h"
```

## Macros

- #define [GPIO\\_DM\\_FIELD\\_SIZE](#) (3u)
- #define [GPIO\\_DM\\_FIELD\\_MASK](#) (7u)
- #define [GPIO\\_INT\\_FIELD\\_MASK](#) (3u)
- #define [HSIOM\\_FIELD\\_SHIFT](#) (2u)
- #define [GPIO\\_PORT0\\_INTR\\_NO](#) (0u)
- #define [GPIO\\_PORT1\\_INTR\\_NO](#) (1u)
- #define [GPIO\\_PORT2\\_INTR\\_NO](#) (2u)
- #define [GPIO\\_PORT3\\_INTR\\_NO](#) (3u)
- #define [GPIO\\_PORT4\\_INTR\\_NO](#) (4u)
- #define [GPIO\\_PORT5\\_INTR\\_NO](#) (5u)
- #define [GPIO\\_PORT6\\_INTR\\_NO](#) (6u)

## Typedefs

- typedef void(\* [gpio\\_intr\\_cb\\_t](#)) ([gpio\\_port\\_pin\\_t](#) port\_pin, bool pin\_state)

## Enumerations

- enum [gpio\\_port\\_pin\\_t](#) {  
[GPIO\\_PORT\\_0\\_PIN\\_0](#) = 0x00, [GPIO\\_PORT\\_0\\_PIN\\_1](#), [GPIO\\_PORT\\_0\\_PIN\\_2](#), [GPIO\\_PORT\\_0\\_PIN\\_3](#),  
[GPIO\\_PORT\\_0\\_PIN\\_4](#), [GPIO\\_PORT\\_0\\_PIN\\_5](#), [GPIO\\_PORT\\_0\\_PIN\\_6](#), [GPIO\\_PORT\\_0\\_PIN\\_7](#),  
[GPIO\\_PORT\\_1\\_PIN\\_0](#) = 0x10, [GPIO\\_PORT\\_1\\_PIN\\_1](#), [GPIO\\_PORT\\_1\\_PIN\\_2](#), [GPIO\\_PORT\\_1\\_PIN\\_3](#),  
[GPIO\\_PORT\\_1\\_PIN\\_4](#), [GPIO\\_PORT\\_1\\_PIN\\_5](#), [GPIO\\_PORT\\_1\\_PIN\\_6](#), [GPIO\\_PORT\\_1\\_PIN\\_7](#),  
[GPIO\\_PORT\\_2\\_PIN\\_0](#) = 0x20, [GPIO\\_PORT\\_2\\_PIN\\_1](#), [GPIO\\_PORT\\_2\\_PIN\\_2](#), [GPIO\\_PORT\\_2\\_PIN\\_3](#),  
[GPIO\\_PORT\\_2\\_PIN\\_4](#), [GPIO\\_PORT\\_2\\_PIN\\_5](#), [GPIO\\_PORT\\_2\\_PIN\\_6](#), [GPIO\\_PORT\\_2\\_PIN\\_7](#),  
[GPIO\\_PORT\\_3\\_PIN\\_0](#) = 0x30, [GPIO\\_PORT\\_3\\_PIN\\_1](#), [GPIO\\_PORT\\_3\\_PIN\\_2](#), [GPIO\\_PORT\\_3\\_PIN\\_3](#),  
[GPIO\\_PORT\\_3\\_PIN\\_4](#), [GPIO\\_PORT\\_3\\_PIN\\_5](#), [GPIO\\_PORT\\_3\\_PIN\\_6](#), [GPIO\\_PORT\\_3\\_PIN\\_7](#),  
[GPIO\\_PORT\\_4\\_PIN\\_0](#) = 0x40, [GPIO\\_PORT\\_4\\_PIN\\_1](#), [GPIO\\_PORT\\_4\\_PIN\\_2](#), [GPIO\\_PORT\\_4\\_PIN\\_3](#),  
[GPIO\\_PORT\\_4\\_PIN\\_4](#), [GPIO\\_PORT\\_4\\_PIN\\_5](#), [GPIO\\_PORT\\_4\\_PIN\\_6](#), [GPIO\\_PORT\\_4\\_PIN\\_7](#),  
[GPIO\\_PORT\\_5\\_PIN\\_0](#) = 0x50, [GPIO\\_PORT\\_5\\_PIN\\_1](#), [GPIO\\_PORT\\_5\\_PIN\\_2](#), [GPIO\\_PORT\\_5\\_PIN\\_3](#),  
[GPIO\\_PORT\\_5\\_PIN\\_4](#), [GPIO\\_PORT\\_5\\_PIN\\_5](#), [GPIO\\_PORT\\_5\\_PIN\\_6](#), [GPIO\\_PORT\\_5\\_PIN\\_7](#),  
[GPIO\\_PORT\\_6\\_PIN\\_0](#) = 0x60, [GPIO\\_PORT\\_6\\_PIN\\_1](#) }
- enum [gpio\\_dm\\_t](#) {  
[GPIO\\_DM\\_HIZ\\_ANALOG](#) = 0, [GPIO\\_DM\\_HIZ\\_DIGITAL](#), [GPIO\\_DM\\_RES\\_UP](#), [GPIO\\_DM\\_RES\\_DWN](#),  
[GPIO\\_DM\\_OD\\_LOW](#), [GPIO\\_DM\\_OD\\_HIGH](#), [GPIO\\_DM\\_STRONG](#), [GPIO\\_DM\\_RES\\_UPDOWN](#) }
- enum [gpio\\_intr\\_t](#) { [GPIO\\_INTR\\_DISABLE](#) = 0, [GPIO\\_INTR\\_RISING](#), [GPIO\\_INTR\\_FALLING](#), [GPIO\\_INTR\\_BOTH](#) }
- enum [hsiom\\_mode\\_t](#) { [HSIOM\\_MODE\\_GPIO](#) = 0 }

## Functions

- void [gpio\\_set\\_value](#) ([gpio\\_port\\_pin\\_t](#) port\_pin, bool value)
- bool [gpio\\_read\\_value](#) ([gpio\\_port\\_pin\\_t](#) port\_pin)
- void [gpio\\_set\\_drv\\_mode](#) ([gpio\\_port\\_pin\\_t](#) port\_pin, [gpio\\_dm\\_t](#) drv\_mode)
- void [gpio\\_int\\_set\\_config](#) ([gpio\\_port\\_pin\\_t](#) port\_pin, [uint8\\_t](#) int\_mode)
- [ccg\\_status\\_t](#) [gpio\\_register\\_intr\\_cb](#) ([uint8\\_t](#) port, [gpio\\_intr\\_cb\\_t](#) intr\_cb)
- void [hsiom\\_set\\_config](#) ([gpio\\_port\\_pin\\_t](#) port\_pin, [hsiom\\_mode\\_t](#) hsiom\_mode)
- void [gpio\\_hsiom\\_set\\_config](#) ([gpio\\_port\\_pin\\_t](#) port\_pin, [hsiom\\_mode\\_t](#) hsiom\_mode, [gpio\\_dm\\_t](#) drv\_mode, bool value)

- void `gpio_set_lvttl_mode` (uint8\_t port)
- bool `gpio_get_intr` (gpio\_port\_pin\_t port\_pin)
- void `gpio_clear_intr` (gpio\_port\_pin\_t port\_pin)

### 6.32.1 Detailed Description

GPIO and IO mapping control functions.

### 6.32.2 Macro Definition Documentation

#### 6.32.2.1 GPIO\_DM\_FIELD\_MASK

```
#define GPIO_DM_FIELD_MASK (7u)
```

GPIO drive mode field mask without offset.

#### 6.32.2.2 GPIO\_DM\_FIELD\_SIZE

```
#define GPIO_DM_FIELD_SIZE (3u)
```

GPIO drive mode field size.

#### 6.32.2.3 GPIO\_INT\_FIELD\_MASK

```
#define GPIO_INT_FIELD_MASK (3u)
```

GPIO interrupt configuration field mask.

#### 6.32.2.4 GPIO\_PORT0\_INTR\_NO

```
#define GPIO_PORT0_INTR_NO (0u)
```

Interrupt vector number for P0.x pins.

#### 6.32.2.5 GPIO\_PORT1\_INTR\_NO

```
#define GPIO_PORT1_INTR_NO (1u)
```

Interrupt vector number for P1.x pins.

#### 6.32.2.6 GPIO\_PORT2\_INTR\_NO

```
#define GPIO_PORT2_INTR_NO (2u)
```

Interrupt vector number for P2.x pins. Only valid if P2.x pins are present.

#### 6.32.2.7 GPIO\_PORT3\_INTR\_NO

```
#define GPIO_PORT3_INTR_NO (3u)
```

Interrupt vector number for P3.x pins. Only valid if P3.x pins are present.



### 6.32.2.8 GPIO\_PORT4\_INTR\_NO

```
#define GPIO_PORT4_INTR_NO (4u)
```

Interrupt vector number for P4.x pins. Only valid if P4.x pins are present.

### 6.32.2.9 GPIO\_PORT5\_INTR\_NO

```
#define GPIO_PORT5_INTR_NO (5u)
```

Interrupt vector number for P5.x pins. Only valid if P5.x pins are present.

### 6.32.2.10 GPIO\_PORT6\_INTR\_NO

```
#define GPIO_PORT6_INTR_NO (6u)
```

Interrupt vector number for P6.x pins. Only valid if P6.x pins are present.

### 6.32.2.11 HSIOM\_FIELD\_SHIFT

```
#define HSIOM_FIELD_SHIFT (2u)
```

HSIOM configuration field size as left shift value.

## 6.32.3 Typedef Documentation

### 6.32.3.1 gpio\_intr\_cb\_t

```
typedef void(* gpio_intr_cb_t) (gpio_port_pin_t port_pin, bool pin_state)
```

Callback type for GPIO interrupt notification.

#### Parameters

|                  |                                                 |
|------------------|-------------------------------------------------|
| <i>port_pin</i>  | Parameter identifying the GPIO port and pin.    |
| <i>pin_state</i> | State of the GPIO when at the beginning of ISR. |

## 6.32.4 Enumeration Type Documentation

### 6.32.4.1 gpio\_dm\_t

```
enum gpio_dm_t
```

Various GPIO drive modes supported by the CCGx devices.

This enumeration lists the various drive modes supported by CCGx IOs which are configured as GPIOs. Please refer to `hsiom_mode_t` for the IO mapping settings available for each CCGx IO.

See also

[hsiom\\_mode\\_t](#)

## Enumerator

|                     |                                           |
|---------------------|-------------------------------------------|
| GPIO_DM_HIZ_ANALOG  | Output buffer off (HiZ), input buffer off |
| GPIO_DM_HIZ_DIGITAL | Output buffer off (HiZ), input buffer on  |
| GPIO_DM_RES_UP      | Resistive pull-up                         |
| GPIO_DM_RES_DWN     | Resistive pull-down                       |
| GPIO_DM_OD_LOW      | Drive low and HZI for high                |
| GPIO_DM_OD_HIGH     | Drive high and HZI for low                |
| GPIO_DM_STRONG      | Strong low and high drive                 |
| GPIO_DM_RES_UPDOWN  | Resistive pull-up and pull-down           |

## 6.32.4.2 gpio\_intr\_t

enum `gpio_intr_t`

Various GPIO interrupt modes supported by the device.

## Enumerator

|                   |                                             |
|-------------------|---------------------------------------------|
| GPIO_INTR_DISABLE | GPIO interrupt disabled.                    |
| GPIO_INTR_RISING  | Interrupt on rising edge of input.          |
| GPIO_INTR_FALLING | Interrupt on falling edge of input.         |
| GPIO_INTR_BOTH    | Interrupt on both rising and falling edges. |

## 6.32.4.3 gpio\_port\_pin\_t

enum `gpio_port_pin_t`

List of pins supported on CCGx devices.

This enumeration lists the port and pin assignment for the pins available on the CCGx USB-PD controllers. This is a superset of all pins supported across all of the devices in the family, and includes pins such as CC1/CC2 which have fixed functionality.

Please refer to the respective device datasheets to identify the pins that can be used as GPIOs.

## Enumerator

|                        |                      |
|------------------------|----------------------|
| GPIO_PORT_0_PIN↔<br>_0 | P0.0: Port 0, Pin 0. |
| GPIO_PORT_0_PIN↔<br>_1 | P0.1: Port 0, Pin 1. |
| GPIO_PORT_0_PIN↔<br>_2 | P0.2: Port 0, Pin 2. |
| GPIO_PORT_0_PIN↔<br>_3 | P0.3: Port 0, Pin 3. |
| GPIO_PORT_0_PIN↔<br>_4 | P0.4: Port 0, Pin 4. |
| GPIO_PORT_0_PIN↔<br>_5 | P0.5: Port 0, Pin 5. |

## Enumerator

|                        |                      |
|------------------------|----------------------|
| GPIO_PORT_0_PIN↔<br>_6 | P0.6: Port 0, Pin 6. |
| GPIO_PORT_0_PIN↔<br>_7 | P0.7: Port 0, Pin 7. |
| GPIO_PORT_1_PIN↔<br>_0 | P1.0: Port 1, Pin 0. |
| GPIO_PORT_1_PIN↔<br>_1 | P1.1: Port 1, Pin 1. |
| GPIO_PORT_1_PIN↔<br>_2 | P1.2: Port 1, Pin 2. |
| GPIO_PORT_1_PIN↔<br>_3 | P1.3: Port 1, Pin 3. |
| GPIO_PORT_1_PIN↔<br>_4 | P1.4: Port 1, Pin 4. |
| GPIO_PORT_1_PIN↔<br>_5 | P1.5: Port 1, Pin 5. |
| GPIO_PORT_1_PIN↔<br>_6 | P1.6: Port 1, Pin 6. |
| GPIO_PORT_1_PIN↔<br>_7 | P1.7: Port 1, Pin 7. |
| GPIO_PORT_2_PIN↔<br>_0 | P2.0: Port 2, Pin 0. |
| GPIO_PORT_2_PIN↔<br>_1 | P2.1: Port 2, Pin 1. |
| GPIO_PORT_2_PIN↔<br>_2 | P2.2: Port 2, Pin 2. |
| GPIO_PORT_2_PIN↔<br>_3 | P2.3: Port 2, Pin 3. |
| GPIO_PORT_2_PIN↔<br>_4 | P2.4: Port 2, Pin 4. |
| GPIO_PORT_2_PIN↔<br>_5 | P2.5: Port 2, Pin 5. |
| GPIO_PORT_2_PIN↔<br>_6 | P2.6: Port 2, Pin 6. |
| GPIO_PORT_2_PIN↔<br>_7 | P2.7: Port 2, Pin 7. |
| GPIO_PORT_3_PIN↔<br>_0 | P3.0: Port 3, Pin 0. |
| GPIO_PORT_3_PIN↔<br>_1 | P3.1: Port 3, Pin 1. |
| GPIO_PORT_3_PIN↔<br>_2 | P3.2: Port 3, Pin 2. |
| GPIO_PORT_3_PIN↔<br>_3 | P3.3: Port 3, Pin 3. |
| GPIO_PORT_3_PIN↔<br>_4 | P3.4: Port 3, Pin 4. |
| GPIO_PORT_3_PIN↔<br>_5 | P3.5: Port 3, Pin 5. |
| GPIO_PORT_3_PIN↔<br>_6 | P3.6: Port 3, Pin 6. |
| GPIO_PORT_3_PIN↔<br>_7 | P3.7: Port 3, Pin 7. |

## Enumerator

|                        |                      |
|------------------------|----------------------|
| GPIO_PORT_4_PIN↔<br>_0 | P4.0: Port 4, Pin 0. |
| GPIO_PORT_4_PIN↔<br>_1 | P4.1: Port 4, Pin 1. |
| GPIO_PORT_4_PIN↔<br>_2 | P4.2: Port 4, Pin 2. |
| GPIO_PORT_4_PIN↔<br>_3 | P4.3: Port 4, Pin 3. |
| GPIO_PORT_4_PIN↔<br>_4 | P4.4: Port 4, Pin 4. |
| GPIO_PORT_4_PIN↔<br>_5 | P4.5: Port 4, Pin 5. |
| GPIO_PORT_4_PIN↔<br>_6 | P4.6: Port 4, Pin 6. |
| GPIO_PORT_4_PIN↔<br>_7 | P4.7: Port 4, Pin 7. |
| GPIO_PORT_5_PIN↔<br>_0 | P5.0: Port 5, Pin 0. |
| GPIO_PORT_5_PIN↔<br>_1 | P5.1: Port 5, Pin 1. |
| GPIO_PORT_5_PIN↔<br>_2 | P5.2: Port 5, Pin 2. |
| GPIO_PORT_5_PIN↔<br>_3 | P5.3: Port 5, Pin 3. |
| GPIO_PORT_5_PIN↔<br>_4 | P5.4: Port 5, Pin 4. |
| GPIO_PORT_5_PIN↔<br>_5 | P5.5: Port 5, Pin 5. |
| GPIO_PORT_5_PIN↔<br>_6 | P5.6: Port 5, Pin 6. |
| GPIO_PORT_5_PIN↔<br>_7 | P5.7: Port 5, Pin 7. |
| GPIO_PORT_6_PIN↔<br>_0 | P6.0: Port 6, Pin 0. |
| GPIO_PORT_6_PIN↔<br>_1 | P6.1: Port 6, Pin 1. |

## 6.32.4.4 hsiom\_mode\_t

```
enum hsiom_mode_t
```

Various IO matrix configuration modes.

Most of the IOs on CCGx devices & DMC (Dock Management Controller), (except fixed function IOs like Vdd, G↔ND, CCx), can be configured to select one from many available options. This enumerated type lists the various IO functions that can be selected for the flexible IOs. Not all functionality can be configured to any selected IO. The selectable functionality for each IO is fixed and cannot be altered. For example, in case of CCG3 or DMC, SWD\_CLK cannot be configured to any other pin other than P2.1. But P2.1 can be configured to function as GPIO, SCB1\_SPI\_MOSI, SCB1\_I2C\_SDA or SCB1\_UART\_RTS.

Please refer to the respective device datasheets for more information about these IO options.

## Enumerator

|                 |                                           |
|-----------------|-------------------------------------------|
| HSIOM_MODE_GPIO | Special functions disabled. Used as GPIO. |
|-----------------|-------------------------------------------|

## 6.32.5 Function Documentation

### 6.32.5.1 gpio\_clear\_intr()

```
void gpio_clear_intr (
 gpio_port_pin_t port_pin)
```

Clear interrupt status on the specified GPIO.

This function clears any active interrupts on the specified GPIO.

## Parameters

|                 |                    |
|-----------------|--------------------|
| <i>port_pin</i> | Pin to be updated. |
|-----------------|--------------------|

## Returns

None

### 6.32.5.2 gpio\_get\_intr()

```
bool gpio_get_intr (
 gpio_port_pin_t port_pin)
```

Read the interrupt status on a specific GPIO.

This function checks whether there are any active interrupts on the specified GPIO.

## Parameters

|                 |                    |
|-----------------|--------------------|
| <i>port_pin</i> | Pin to be queried. |
|-----------------|--------------------|

## Returns

true if interrupt is active, false otherwise.

### 6.32.5.3 gpio\_hsiom\_set\_config()

```
void gpio_hsiom_set_config (
 gpio_port_pin_t port_pin,
 hsiom_mode_t hsiom_mode,
 gpio_dm_t drv_mode,
 bool value)
```

Single function for complete configuration of a CCG IO.

This is a single API which can be used to configure the IO mode, the drive mode and the current value of a CCG device IO. No error checks are performed, and the caller should ensure that the settings provided are valid for the selected IO.

#### Parameters

|                   |                       |
|-------------------|-----------------------|
| <i>port_pin</i>   | Pin to be configured. |
| <i>hsiom_mode</i> | Desired IO mode.      |
| <i>drv_mode</i>   | Desired drive mode.   |
| <i>value</i>      | Desired output state. |

#### Returns

None

#### 6.32.5.4 gpio\_int\_set\_config()

```
void gpio_int_set_config (
 gpio_port_pin_t port_pin,
 uint8_t int_mode)
```

Configure the GPIO with the desired interrupt setting.

This API configures the interrupt mode for the specified GPIO. It does not do any error check and will just configure the GPIO interrupt setting. Care should be taken to make sure that wrong indexing is not done.

#### Parameters

|                 |                         |
|-----------------|-------------------------|
| <i>port_pin</i> | Pin to be configured.   |
| <i>int_mode</i> | Desired interrupt mode. |

#### Returns

None

#### 6.32.5.5 gpio\_read\_value()

```
bool gpio_read_value (
 gpio_port_pin_t port_pin)
```

Get the GPIO current state.

This API retrieves the current state of a pin, assuming it has been configured as a GPIO. It is the caller's responsibility to ensure that the HSIOM and drive mode settings for the pin have been set correctly.

The API does not do any error check and will just read the GPIO state register. Care should be taken to make sure that wrong indexing is not done.

#### Parameters

|                 |                    |
|-----------------|--------------------|
| <i>port_pin</i> | Pin to be queried. |
|-----------------|--------------------|

**Returns**

Current state of the pin.

**See also**

[hsiom\\_set\\_config](#)  
[gpio\\_set\\_drv\\_mode](#)

**6.32.5.6 gpio\_register\_intr\_cb()**

```
ccg_status_t gpio_register_intr_cb (
 uint8_t port,
 gpio_intr_cb_t intr_cb)
```

Register a callback to be called when a GPIO interrupt associated with a specific port is triggered. The `gpio_int_↔set_config` should be used to configure the desired interrupts for each of the relevant pins.

**Parameters**

|                |                                                         |
|----------------|---------------------------------------------------------|
| <i>port</i>    | GPIO port on which interrupts are to be updated.        |
| <i>intr_cb</i> | Pointer to callback function to be called on interrupt. |

**6.32.5.7 gpio\_set\_drv\_mode()**

```
void gpio_set_drv_mode (
 gpio_port_pin_t port_pin,
 gpio_dm_t drv_mode)
```

Configure the GPIO with the desired drive mode.

This API updates the drive mode for the selected CCG device IO. The API does not do any error check and will just set the drive mode of GPIO. The caller should ensure that the HSIOM setting for the pin has been selected correctly.

**Parameters**

|                 |                       |
|-----------------|-----------------------|
| <i>port_pin</i> | Pin to be configured. |
| <i>drv_mode</i> | Desired drive mode.   |

**Returns**

None

**6.32.5.8 gpio\_set\_lvttl\_mode()**

```
void gpio_set_lvttl_mode (
 uint8_t port)
```

Set the input buffer voltage for a port to LVTTTL.



This API sets the input buffer voltage for a complete CCG device port to LVTTL levels. This is required to be set for ports that include the IOs used for the I2C based HPI interface.

#### Parameters

|             |                                            |
|-------------|--------------------------------------------|
| <i>port</i> | IO port to be configured for LVTTL levels. |
|-------------|--------------------------------------------|

#### Returns

None

#### 6.32.5.9 gpio\_set\_value()

```
void gpio_set_value (
 gpio_port_pin_t port_pin,
 bool value)
```

Sets the GPIO to the required state.

This function updates the output state of a GPIO pin. The API does not do any error check and will just update the output state. It is the caller's responsibility to ensure that the HSIOM and drive mode settings for the pin have been selected as required.

Care should be taken to make sure that wrong indexing is not done.

#### Parameters

|                 |                                           |
|-----------------|-------------------------------------------|
| <i>port_pin</i> | Pin to be updated.                        |
| <i>value</i>    | Value to drive on the pin: 0=LOW, 1=HIGH. |

#### Returns

None

#### See also

[hsiom\\_set\\_config](#)  
[gpio\\_set\\_drv\\_mode](#)

#### 6.32.5.10 hsiom\_set\_config()

```
void hsiom_set_config (
 gpio_port_pin_t port_pin,
 hsiom_mode_t hsiom_mode)
```

Select the IO mode for a CCG device IO.

This API configures the IO mode for a CCG device IO. It does not do any error check and will just set the HSIOM configuration for the GPIO. Care should be taken to make sure that wrong indexing is not done.

#### Parameters

|                   |                       |
|-------------------|-----------------------|
| <i>port_pin</i>   | Pin to be configured. |
| <i>hsiom_mode</i> | Desired IO mode.      |

**Returns**

None

**See also**[hsiom\\_mode\\_t](#)**6.33 system/instrumentation.h File Reference**

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

**Typedefs**

- typedef void(\* [instrumentation\\_cb\\_t](#)) (uint8\_t port, uint8\_t evt)

**Enumerations**

- enum [inst\\_evt\\_t](#) { [INST\\_EVT\\_WDT\\_RESET](#) = 0, [INST\\_EVT\\_HARD\\_FAULT](#) = 1 }

**Functions**

- void [instrumentation\\_init](#) (void)
- void [instrumentation\\_start](#) (void)
- void [instrumentation\\_task](#) (void)
- void [instrumentation\\_register\\_cb](#) ([instrumentation\\_cb\\_t](#) cb)

**6.33.1 Detailed Description**

Header file containing application instrumentation definitions.

This USB-PD port controller application supports high level instrumentation to track the task execution latencies and runtime stack usage. This header file contains the definitions associated with these functions.

**6.33.2 Enumeration Type Documentation****6.33.2.1 inst\_evt\_t**

```
enum inst_evt_t
```

Enumeration of all instrumentation fault events.

**Enumerator**

|                                     |                                                       |
|-------------------------------------|-------------------------------------------------------|
| <a href="#">INST_EVT_WDT_RESET</a>  | 0x00: Instrumentation fault event for watchdog reset. |
| <a href="#">INST_EVT_HARD_FAULT</a> | 0x01: Instrumentation fault event for hard fault.     |

### 6.33.3 Function Documentation

#### 6.33.3.1 instrumentation\_init()

```
void instrumentation_init (
 void)
```

Initialize data structures associated with application instrumentation.

#### Returns

None

#### 6.33.3.2 instrumentation\_start()

```
void instrumentation_start (
 void)
```

Start any timers or tasks associated with application instrumentation.

#### Returns

None

## 6.34 system/srom.h File Reference

```
#include <config.h>
#include <stdio.h>
#include <string.h>
#include "srom_config.h"
#include "srom_vars_default.h"
```

### Macros

- #define [CALL\\_MAP](#)(str) (str)

#### 6.34.1 Detailed Description

SROM Code header file.

#### 6.34.2 Macro Definition Documentation

##### 6.34.2.1 CALL\_MAP

```
#define CALL_MAP(
 str) (str)
```

This file shall be updated during ROM code base updates only.

## 6.35 system/status.h File Reference

### Macros

- `#define CCG_STATUS_CODE_OFFSET (2)`
- `#define CCG_STATUS_TO_HPI_RESPONSE(c) ((c) + CCG_STATUS_CODE_OFFSET)`

### Enumerations

- enum `ccg_status_t` {  
`CCG_STAT_NO_RESPONSE = -2, CCG_STAT_SUCCESS = 0, CCG_STAT_FLASH_DATA_AVAILABLE,`  
`CCG_STAT_BAD_PARAM,`  
`CCG_STAT_INVALID_COMMAND = 3, CCG_STAT_FLASH_UPDATE_FAILED = 5, CCG_STAT_INVALID_FW,`  
`CCG_STAT_INVALID_ARGUMENT,`  
`CCG_STAT_NOT_SUPPORTED, CCG_STAT_INVALID_SIGNATURE, CCG_STAT_TRANS_FAILURE,`  
`CCG_STAT_CMD_FAILURE,`  
`CCG_STAT_FAILURE, CCG_STAT_READ_DATA, CCG_STAT_NOT_READY, CCG_STAT_BUSY,`  
`CCG_STAT_TIMEOUT, CCG_STAT_INVALID_PORT }`

#### 6.35.1 Detailed Description

API return status definitions.

#### 6.35.2 Macro Definition Documentation

##### 6.35.2.1 CCG\_STATUS\_CODE\_OFFSET

```
#define CCG_STATUS_CODE_OFFSET (2)
```

Value to be added to the status code to get the HPI/UVDM response code.

The HPI/UVDM interface response codes use a different convention than the core function return codes. This value represents the offset that should be added to the function return code to get the HPI/UVDM response code.

##### 6.35.2.2 CCG\_STATUS\_TO\_HPI\_RESPONSE

```
#define CCG_STATUS_TO_HPI_RESPONSE(
 c) ((c) + CCG_STATUS_CODE_OFFSET)
```

Convert CCG status code to HPI/UVDM response code.

#### 6.35.3 Enumeration Type Documentation

##### 6.35.3.1 ccg\_status\_t

```
enum ccg_status_t
```

Interface status codes.

Enumeration to hold status codes for all CCG interfaces. These values are pre-defined for each interface and should not be modified. To make interface usage easier, the enumeration starts at -2. This allows the success status to

have a value of zero. The response code should be incremented by two before sending out on the individual interfaces.

#### Enumerator

|                               |                                                         |
|-------------------------------|---------------------------------------------------------|
| CCG_STAT_NO_RESPONSE          | Special status code indicating no response.             |
| CCG_STAT_SUCCESS              | Success status.                                         |
| CCG_STAT_FLASH_DATA_AVAILABLE | Special status code indicating flash data availability. |
| CCG_STAT_BAD_PARAM            | Bad input parameter.                                    |
| CCG_STAT_INVALID_COMMAND      | Operation failed due to invalid command.                |
| CCG_STAT_FLASH_UPDATE_FAILED  | Flash write operation failed.                           |
| CCG_STAT_INVALID_FW           | Special status code indicating invalid firmware         |
| CCG_STAT_INVALID_ARGUMENT     | Operation failed due to invalid arguments.              |
| CCG_STAT_NOT_SUPPORTED        | Feature not supported.                                  |
| CCG_STAT_INVALID_SIGNATURE    | Invalid signature parameter identified.                 |
| CCG_STAT_TRANS_FAILURE        | Transaction failure status.                             |
| CCG_STAT_CMD_FAILURE          | Command failure status                                  |
| CCG_STAT_FAILURE              | Generic failure status.                                 |
| CCG_STAT_READ_DATA            | Special status code indicating read data availability.  |
| CCG_STAT_NOT_READY            | Operation failed due to device/stack not ready.         |
| CCG_STAT_BUSY                 | Operation failed due to device/stack busy status.       |
| CCG_STAT_TIMEOUT              | Operation timed out.                                    |
| CCG_STAT_INVALID_PORT         | Invalid port number.                                    |

## 6.36 system/system.h File Reference

```
#include "stdint.h"
```

#### Macros

- #define [SYS\\_BOOT\\_VERSION\\_ADDRESS](#) (0x000000E0)
- #define [SYS\\_FW\\_VERSION\\_OFFSET](#) (0x000000E0)
- #define [SYS\\_APP\\_VERSION\\_OFFSET](#) (0x00000004)
- #define [SYS\\_SILICON\\_ID\\_OFFSET](#) (0x000000EA)
- #define [SYS\\_BOOT\\_TYPE\\_FIELD\\_OFFSET](#) (0x000000EC)
- #define [SYS\\_FW\\_CUSTOM\\_INFO\\_OFFSET](#) (0x000000C0)
- #define [SYS\\_METADATA\\_VALID\\_SIG](#) (0x4359)
- #define [SYS\\_PSEUDO\\_METADATA\\_VALID\\_SIG](#) (0x4350)
- #define [SYS\\_BOOT\\_MODE\\_RQT\\_SIG](#) (0x424C)
- #define [SYS\\_CONFIG\\_TABLE\\_SIGN](#) (0x4359u)
- #define [SYS\\_INVALID\\_FW\\_START\\_ADDR](#) (0x00000000)
- #define [SYS\\_BOOT\\_TYPE\\_SECURE\\_BOOT\\_MASK](#) (0x01)
- #define [SYS\\_BOOT\\_TYPE\\_FW\\_UPDATE\\_INTERFACE\\_POS](#) (0x01)
- #define [SYS\\_BOOT\\_TYPE\\_FW\\_UPDATE\\_INTERFACE\\_MASK](#) (0x02)
- #define [SYS\\_BOOT\\_TYPE\\_APP\\_PRIORITY\\_POS](#) (0x02)
- #define [SYS\\_SILICON\\_ID\\_MASK](#) (0xFF00)

## Enumerations

- enum `sys_fw_mode_t` { `SYS_FW_MODE_BOOTLOADER` = 0, `SYS_FW_MODE_FWIMAGE_1`, `SYS_FW_MODE_FWIMAGE_2`, `SYS_FW_MODE_INVALID` }

## Functions

- void `sys_set_device_mode` (`sys_fw_mode_t` fw\_mode)
- `sys_fw_mode_t` `sys_get_device_mode` (void)
- uint8\_t \* `sys_get_boot_version` (void)
- uint8\_t \* `sys_get_img1_fw_version` (void)
- uint8\_t \* `sys_get_img2_fw_version` (void)
- uint32\_t `sys_get_fw_img1_start_addr` (void)
- uint32\_t `sys_get_fw_img2_start_addr` (void)
- uint8\_t `sys_get_recent_fw_image` (void)
- void `sys_get_silicon_id` (uint32\_t \*silicon\_id)
- uint8\_t `get_silicon_revision` (void)
- uint32\_t `sys_get_custom_info_addr` (void)
- uint16\_t `sys_get_bcdDevice_version` (uint32\_t ver\_addr)

## Variables

- `sys_fw_mode_t` `gl_active_fw`
- uint8\_t `gl_invalid_version` [8]

### 6.36.1 Detailed Description

Support functions and definitions for bootloader and flash updates.

### 6.36.2 Macro Definition Documentation

#### 6.36.2.1 SYS\_APP\_VERSION\_OFFSET

```
#define SYS_APP_VERSION_OFFSET (0x00000004)
```

Offset of App version from start of firmware version.

#### 6.36.2.2 SYS\_BOOT\_MODE\_RQT\_SIG

```
#define SYS_BOOT_MODE_RQT_SIG (0x424C)
```

Boot Mode Request Signature: "BL"

#### 6.36.2.3 SYS\_BOOT\_TYPE\_APP\_PRIORITY\_POS

```
#define SYS_BOOT_TYPE_APP_PRIORITY_POS (0x02)
```

Position of app priority bit in Bootloader type.

#### 6.36.2.4 SYS\_BOOT\_TYPE\_FIELD\_OFFSET

```
#define SYS_BOOT_TYPE_FIELD_OFFSET (0x000000EC)
```

Offset of Bootloader type in Bootloader flash region.

#### 6.36.2.5 SYS\_BOOT\_TYPE\_FW\_UPDATE\_INTERFACE\_MASK

```
#define SYS_BOOT_TYPE_FW_UPDATE_INTERFACE_MASK (0x02)
```

Mask to get FW update interface bit in BL type.

#### 6.36.2.6 SYS\_BOOT\_TYPE\_FW\_UPDATE\_INTERFACE\_POS

```
#define SYS_BOOT_TYPE_FW_UPDATE_INTERFACE_POS (0x01)
```

Position of FW update interface bit in BL type.

#### 6.36.2.7 SYS\_BOOT\_TYPE\_SECURE\_BOOT\_MASK

```
#define SYS_BOOT_TYPE_SECURE_BOOT_MASK (0x01)
```

Mask to get Secure Boot feature bit in BL type.

#### 6.36.2.8 SYS\_BOOT\_VERSION\_ADDRESS

```
#define SYS_BOOT_VERSION_ADDRESS (0x000000E0)
```

Boot loader version address in FLASH.

#### 6.36.2.9 SYS\_CONFIG\_TABLE\_SIGN

```
#define SYS_CONFIG_TABLE_SIGN (0x4359u)
```

Configuration table valid signature: "CY"

#### 6.36.2.10 SYS\_FW\_CUSTOM\_INFO\_OFFSET

```
#define SYS_FW_CUSTOM_INFO_OFFSET (0x000000C0)
```

Offset of Customer Specific Info from FW start.

#### 6.36.2.11 SYS\_FW\_VERSION\_OFFSET

```
#define SYS_FW_VERSION_OFFSET (0x000000E0)
```

Offset of FW version from start of FW image in flash.

#### 6.36.2.12 SYS\_INVALID\_FW\_START\_ADDR

```
#define SYS_INVALID_FW_START_ADDR (0x00000000)
```

Invalid FW Start Address.

**6.36.2.13 SYS\_METADATA\_VALID\_SIG**

```
#define SYS_METADATA_VALID_SIG (0x4359)
```

Metadata table valid signature: "CY"

**6.36.2.14 SYS\_PSEUDO\_METADATA\_VALID\_SIG**

```
#define SYS_PSEUDO_METADATA_VALID_SIG (0x4350)
```

Pseudo-Metadata valid signature: "CP"

**6.36.2.15 SYS\_SILICON\_ID\_MASK**

```
#define SYS_SILICON_ID_MASK (0xFF00)
```

Mask to extract the device ID from Silicon ID.

**6.36.2.16 SYS\_SILICON\_ID\_OFFSET**

```
#define SYS_SILICON_ID_OFFSET (0x000000EA)
```

Offset of Silicon ID stored in FW image in flash.

**6.36.3 Enumeration Type Documentation****6.36.3.1 sys\_fw\_mode\_t**

```
enum sys_fw_mode_t
```

List of CCG firmware modes.

Enumerator

|                        |                   |
|------------------------|-------------------|
| SYS_FW_MODE_BOOTLOADER | Bootloader mode.  |
| SYS_FW_MODE_FWIMAGE_1  | Firmware Image #1 |
| SYS_FW_MODE_FWIMAGE_2  | Firmware Image #2 |
| SYS_FW_MODE_INVALID    | Invalid value.    |

**6.36.4 Function Documentation****6.36.4.1 get\_silicon\_revision()**

```
uint8_t get_silicon_revision (
 void)
```

Returns Silicon revision.



**Returns**

Silicon revision B[7:4] - Major rev B[3:0] - Minor rev

**6.36.4.2 sys\_get\_bcdDevice\_version()**

```
uint16_t sys_get_bcdDevice_version (
 uint32_t ver_addr)
```

Get bcdDevice version of device.

This function returns bcdDevice version for the device which can be used as part of D\_ID response, secure boot checks etc. Format of bcdDevice version is documented in the function body.

**Parameters**

|                 |                                    |
|-----------------|------------------------------------|
| <i>ver_addr</i> | Offset of version in Flash memory. |
|-----------------|------------------------------------|

**Returns**

16 bits bcdDevice version.

**6.36.4.3 sys\_get\_boot\_version()**

```
uint8_t* sys_get_boot_version (
 void)
```

Get bootloader version.

The bootloader version is stored at absolute address `SYS_CCG_BOOT_VERSION_ADDRESS` in device FLASH. This function returns a pointer to this version information.

**Returns**

Pointer to the bootloader version information.

**6.36.4.4 sys\_get\_custom\_info\_addr()**

```
uint32_t sys_get_custom_info_addr (
 void)
```

Get start address of Customer info section.

This function returns the start address of Customer info section.

**Returns**

Address of Customer info section.

#### 6.36.4.5 sys\_get\_device\_mode()

```
sys_fw_mode_t sys_get_device_mode (
 void)
```

Get the current firmware mode.

This function retrieves the current firmware mode of the CCG device.

##### Returns

The current firmware mode.

#### 6.36.4.6 sys\_get\_fw\_img1\_start\_addr()

```
uint32_t sys_get_fw_img1_start_addr (
 void)
```

Get the flash start address of firmware image-1.

This function returns the flash address from where firmware image-1 (FW1) has been stored.

##### Returns

Start address of firmware image-1.

#### 6.36.4.7 sys\_get\_fw\_img2\_start\_addr()

```
uint32_t sys_get_fw_img2_start_addr (
 void)
```

Get the flash start address of firmware image-2.

This function returns the flash address from where firmware image-2 (FW2) has been stored.

##### Returns

Start address of firmware image-2.

#### 6.36.4.8 sys\_get\_img1\_fw\_version()

```
uint8_t* sys_get_img1_fw_version (
 void)
```

Get version for firmware image-1.

This function returns a pointer to the version information for firmware image-1 (FW1). The version is located at a fixed offset of CY\_PD\_FW\_VERSION\_OFFSET bytes from the start of the firmware binary.

##### Returns

Pointer to the firmware image-1 version information.

## 6.36.4.9 sys\_get\_img2\_fw\_version()

```
uint8_t* sys_get_img2_fw_version (
 void)
```

Get version for firmware image-2.

This function returns a pointer to the version information for firmware image-2 (FW2). The version is located at a fixed offset of CY\_PD\_FW\_VERSION\_OFFSET bytes from the start of the firmware binary.

**Returns**

Pointer to the firmware image-2 version information.

## 6.36.4.10 sys\_get\_recent\_fw\_image()

```
uint8_t sys_get_recent_fw_image (
 void)
```

Determines the more recently update firmware image.

The CCG Bootloader uses this function to determine the more recently updated firmware image (from among FW1 and FW2) by comparing the sequence numbers of images which are stored in the firmware metadata table. The bootloader loads the most recently updated binary by default (even if its version is older than that of the other firmware binary).

**Returns**

Firmware id: 1 for Image-1 and 2 for Image-2.

## 6.36.4.11 sys\_get\_silicon\_id()

```
void sys_get_silicon_id (
 uint32_t * silicon_id)
```

Get Silicon ID of device.

This function retrieves the Silicon ID of the CCG device.

**Parameters**

|                   |                                           |
|-------------------|-------------------------------------------|
| <i>silicon_id</i> | Pointer to buffer to hold the Silicon ID. |
|-------------------|-------------------------------------------|

**Returns**

None

## 6.36.4.12 sys\_set\_device\_mode()

```
void sys_set_device_mode (
 sys_fw_mode_t fw_mode)
```

Set the current firmware mode.

This function is used by the start-up logic to store the current firmware mode for the CCGx device.

This should not be used outside of the default start-up logic for the CCGx bootloader and firmware applications.

#### Parameters

|                |                                     |
|----------------|-------------------------------------|
| <i>fw_mode</i> | The active firmware mode to be set. |
|----------------|-------------------------------------|

#### Returns

None

## 6.37 system/timer.h File Reference

```
#include "config.h"
#include "stdint.h"
#include "stdbool.h"
```

### Data Structures

- struct [cpg\\_timer\\_t](#)

### Macros

- #define [TIMER\\_NUM\\_TIMERS](#) (31u)
- #define [TIMER\\_MAX\\_TIMEOUT](#) (0xFFFF)
- #define [TIMER\\_INVALID\\_ID](#) (0xFFu)
- #define [TIMER\\_INVALID\\_INDEX](#) (0xFFu)

### Typedefs

- typedef uint8\_t [timer\\_id\\_t](#)
- typedef void(\* [timer\\_cb\\_t](#)) (uint8\_t instance, [timer\\_id\\_t](#) id)

### Functions

- void [timer\\_init](#) (void)
- bool [timer\\_start](#) (uint8\_t instance, [timer\\_id\\_t](#) id, uint16\_t period, [timer\\_cb\\_t](#) cb)
- bool [timer\\_start\\_wocb](#) (uint8\_t instance, [timer\\_id\\_t](#) id, uint16\_t period)
- void [timer\\_stop](#) (uint8\_t instance, [timer\\_id\\_t](#) id)
- bool [timer\\_is\\_running](#) (uint8\_t instance, [timer\\_id\\_t](#) id)
- bool [timer\\_range\\_enabled](#) (uint8\_t instance, [timer\\_id\\_t](#) low, [timer\\_id\\_t](#) high)
- uint16\_t [timer\\_get\\_count](#) (uint8\_t instance, [timer\\_id\\_t](#) id)
- void [timer\\_stop\\_all](#) (uint8\_t instance)
- void [timer\\_stop\\_range](#) (uint8\_t instance, [timer\\_id\\_t](#) start, [timer\\_id\\_t](#) stop)
- uint8\_t [timer\\_num\\_active](#) (uint8\_t instance)
- void [timer\\_enter\\_sleep](#) (void)
- uint16\_t [timer\\_get\\_multiplier](#) (void)

### 6.37.1 Detailed Description

Soft timer header file.

### 6.37.2 Macro Definition Documentation

#### 6.37.2.1 TIMER\_INVALID\_ID

```
#define TIMER_INVALID_ID (0xFFu)
```

Invalid timer ID value.

#### 6.37.2.2 TIMER\_INVALID\_INDEX

```
#define TIMER_INVALID_INDEX (0xFFu)
```

Invalid timer instance.

#### 6.37.2.3 TIMER\_MAX\_TIMEOUT

```
#define TIMER_MAX_TIMEOUT (0xFFFF)
```

Maximum timeout value in milliseconds supported by the module.

#### 6.37.2.4 TIMER\_NUM\_TIMERS

```
#define TIMER_NUM_TIMERS (31u)
```

Number of soft timers supported per instance of the timer module. The number of instances depends on the number of USB-PD ports supported by the device.

### 6.37.3 Typedef Documentation

#### 6.37.3.1 timer\_cb\_t

```
timer_cb_t
```

Timer callback function.

This callback function is invoked on timer expiry and should be treated as interrupt.

#### 6.37.3.2 timer\_id\_t

```
timer_id_t
```

Timer ID type definition.

This type definition is used to identify software timer objects. The timer ID needs to be unique for each soft timer instance and need to be maintained in the application. To maintain uniqueness, the following timer ID allocation rule is expected to be followed.

- PD and Type-C stack ([pd.h](#)) : 0 - 29 : 30 timers

- Base application stack ([app.h](#)) : 30 - 127 : 98 timers
- HPI module ([hpi.h](#)) : 128 - 135 : 8 timers
- I2C module ([i2c.h](#)) : 136 - 143 : 8 timers
- USB module ([usb.h](#)) : 144 - 151 : 8 timers
- IECS module ([iecs.h](#)) : 152 - 159 : 8 timers
- Reserved : 160 - 191 : 32 timers
- Solution (project directory) : 192 - 254 : 63 timers

## 6.37.4 Function Documentation

### 6.37.4.1 timer\_enter\_sleep()

```
void timer_enter_sleep (
 void)
```

Prepare the timer hardware for entering deep sleep.

This function prepares the timer module and the hardware timer for entering device deep sleep. This must be called prior to entering deep sleep mode.

#### Returns

None

### 6.37.4.2 timer\_get\_count()

```
uint16_t timer_get_count (
 uint8_t instance,
 timer_id_t id)
```

Returns the time remaining for timer expiration.

Returns the time remaining for timer expiration.

#### Parameters

|                 |                                                   |
|-----------------|---------------------------------------------------|
| <i>instance</i> | Instance number for which we are using the timer. |
| <i>id</i>       | Unique timer id.                                  |

#### Returns

Time remaining for expiration of the soft timer.

### 6.37.4.3 timer\_get\_multiplier()

```
uint16_t timer_get_multiplier (
 void)
```

Get the number of LF clock ticks required per ms.

**Returns**

Returns the number of LF clock ticks per ms.

**6.37.4.4 timer\_init()**

```
void timer_init (
 void)
```

Initialize the software timer module.

This function initializes the software timer module. This function initializes the data structures for timer management and enables the hardware timer used for the soft timer implementation.

**Returns**

None

**6.37.4.5 timer\_is\_running()**

```
bool timer_is_running (
 uint8_t instance,
 timer_id_t id)
```

Check whether the specified soft timer is currently running.

**Parameters**

|                 |                                                   |
|-----------------|---------------------------------------------------|
| <i>instance</i> | Instance number for which we are using the timer. |
| <i>id</i>       | Unique timer id.                                  |

**Returns**

true if the timer is running, false otherwise.

**6.37.4.6 timer\_num\_active()**

```
uint8_t timer_num_active (
 uint8_t instance)
```

Returns number of active timers.

**Parameters**

|                 |                           |
|-----------------|---------------------------|
| <i>instance</i> | Instance number to query. |
|-----------------|---------------------------|

**Returns**

Number of active timers on this instance.

**6.37.4.7 timer\_range\_enabled()**

```
bool timer_range_enabled (
 uint8_t instance,
 timer_id_t low,
 timer_id_t high)
```

Check whether any soft timer in the specified range are running.

**Parameters**

|                 |                                                   |
|-----------------|---------------------------------------------------|
| <i>instance</i> | Instance number for which we are using the timer. |
| <i>low</i>      | Lowest soft timer ID to be checked.               |
| <i>high</i>     | Highest soft timer ID to be checked.              |

**Returns**

true if any timer in the specified range is running, false otherwise.

**6.37.4.8 timer\_start()**

```
bool timer_start (
 uint8_t instance,
 timer_id_t id,
 uint16_t period,
 timer_cb_t cb)
```

Start a soft timer.

Start a specific soft timer. All soft timers are one-shot timers which will run until the specified period has elapsed. The timer expiration callback will be called at the end of the period, if one is provided.

**Parameters**

|                 |                                                         |
|-----------------|---------------------------------------------------------|
| <i>instance</i> | Timer instance number for which we are using the timer. |
| <i>id</i>       | Unique timer id                                         |
| <i>period</i>   | Timer period in milliseconds.                           |
| <i>cb</i>       | Timer expiration callback. Can be NULL.                 |

**Returns**

true if the timer is started, false if timer start fails.

**6.37.4.9 timer\_start\_wocb()**

```
bool timer_start_wocb (
```



```
uint8_t instance,
timer_id_t id,
uint16_t period)
```

Start a soft timer.

Start a specific soft timer. All soft timers are one-shot timers which will run until the specified period has elapsed.

#### Parameters

|                 |                                                         |
|-----------------|---------------------------------------------------------|
| <i>instance</i> | Timer instance number for which we are using the timer. |
| <i>id</i>       | Unique timer id                                         |
| <i>period</i>   | Timer period in milliseconds.                           |

#### Returns

true if the timer is started, false if timer start fails.

#### 6.37.4.10 timer\_stop()

```
void timer_stop (
 uint8_t instance,
 timer_id_t id)
```

Stop a soft timer.

Stop a soft timer which is currently running.

#### Parameters

|                 |                                                   |
|-----------------|---------------------------------------------------|
| <i>instance</i> | Instance number for which we are using the timer. |
| <i>id</i>       | Unique timer id.                                  |

#### Returns

None

#### 6.37.4.11 timer\_stop\_all()

```
void timer_stop_all (
 uint8_t instance)
```

Stops all soft timers associated with the instance.

#### Parameters

|                 |                                                    |
|-----------------|----------------------------------------------------|
| <i>instance</i> | Instance number on which timers are to be stopped. |
|-----------------|----------------------------------------------------|

**Returns**

None

**6.37.4.12 timer\_stop\_range()**

```
void timer_stop_range (
 uint8_t instance,
 timer_id_t start,
 timer_id_t stop)
```

This function stops all soft timers with ids in the specified range.

**Parameters**

|                 |                                                         |
|-----------------|---------------------------------------------------------|
| <i>instance</i> | Instance number on which soft timers are to be stopped. |
| <i>start</i>    | Starting timer ID. The value is inclusive.              |
| <i>stop</i>     | Ending timer ID. The value is inclusive.                |

**Returns**

None

**6.38 system/timer\_id.h File Reference**

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

**Enumerations**

- enum `ccg_timer_id_t` {
  - PD\_TIMERS\_START\_ID = 0, PD\_CABLE\_TIMER, PD\_NO\_RESPONSE\_TIMER, PD\_CBL\_DSC\_ID\_TIMER,
  - PD\_CBL\_DELAY\_TIMER, PD\_PHY\_BUSY\_TIMER, PD\_GOOD\_CRC\_TX\_TIMER, PD\_HARD\_RESET\_TX\_TIMER,
  - PD\_VCONN\_SWAP\_INITIATOR\_TIMER, PD\_GENERIC\_TIMER, PD\_PPS\_TIMER, PD\_SINK\_TX\_TIMER,
  - PD\_DATA\_RESET\_COMP\_TIMER, PD\_TIMER\_RESERVED\_13, PD\_TIMERS\_END\_ID, TYPEC\_TIMERS\_START\_ID
  - = 15u,
  - TYPEC\_GENERIC\_TIMER2 = 15u, TYPEC\_GENERIC\_TIMER1, TYPEC\_CC1\_DEBOUNCE\_TIMER,
  - TYPEC\_CC2\_DEBOUNCE\_TIMER,
  - TYPEC\_RD\_DEBOUNCE\_TIMER, TYPEC\_VBUS\_DISCHARGE\_TIMER, TYPEC\_ACTIVITY\_TIMER,
  - TYPEC\_RP\_CHANGE\_TIMER,
  - TYPEC\_TIMER\_RESERVED\_23, TYPEC\_TIMERS\_END\_ID, PD\_OCP\_DEBOUNCE\_TIMER, HPD\_RX\_ACTIVITY\_TIMER,
  - PD\_VCONN\_OCP\_DEBOUNCE\_TIMER, PD\_TIMER\_RESERVED\_28, VBUS\_DISCHARGE\_SCHEDULE\_TIMER,
  - APP\_TIMERS\_START\_ID = 30u,
  - APP\_PSOURCE\_EN\_TIMER = 30u, APP\_PSOURCE\_EN\_MONITOR\_TIMER, APP\_PSOURCE\_EN\_HYS\_TIMER,
  - APP\_PSOURCE\_DIS\_TIMER,
  - APP\_PSOURCE\_DIS\_MONITOR\_TIMER, APP\_PSOURCE\_CF\_TIMER, APP\_PSOURCE\_DIS\_EXT\_DIS\_TIMER,
  - APP\_DB\_SNK\_FET\_DIS\_DELAY\_TIMER,
  - APP\_PSINK\_DIS\_TIMER, APP\_PSINK\_DIS\_MONITOR\_TIMER, APP\_VDM\_BUSY\_TIMER, APP\_AME\_TIMEOUT\_TIMER,
  - APP\_VBUS\_OCP\_OFF\_TIMER, APP\_VBUS\_OVP\_OFF\_TIMER, APP\_VBUS\_UVP\_OFF\_TIMER,
  - APP\_VBUS\_SCP\_OFF\_TIMER,
  - APP\_FAULT\_RECOVERY\_TIMER, APP\_SBU\_DELAYED\_CONNECT\_TIMER, APP\_MUX\_DELAY\_TIMER,
  - APP\_MUX\_POLL\_TIMER,
  - APP\_CBL\_DISC\_TRIGGER\_TIMER, APP\_V5V\_CHANGE\_DEBOUNCE\_TIMER, APP\_VCONN\_RECOVERY\_TIMER,

```

APP_OT_DETECTION_TIMER,
APP_PASC_VALLEY_TIMER_ID, APP_PASC_TASK_TIMER_ID, APP_CHUNKED_MSG_RESP_TIMER,
APP_RESET_VDM_LAYER_TIMER,
APP_TIMER_RESERVED_58, APP_TIMER_RESERVED_59, APP_BB_ON_TIMER, APP_BB_OFF_TIMER,
APP_TIMER_RESERVED_62, APP_INITIATE_SWAP_TIMER, APP_INITIATE_SEND_IRQ_CLEAR_ACK,
UCSI_CONNECT_EVENT_TIMER,
APP_VDM_NOT_SUPPORT_RESP_TIMER_ID, UCSI_DPM_RETRY_TIMER, APP_TIMER_RESERVED_68,
APP_TIMER_RESERVED_69,
APP_BC_TIMERS_START_ID, APP_BC_GENERIC_TIMER1, APP_BC_GENERIC_TIMER2, APP_BC_DP_DM_DEBOUNCE_TIMER,
APP_BC_DETACH_DETECT_TIMER, APP_CDP_DP_DM_POLL_TIMER, APP_TIMER_RESERVED_76,
APP_TIMER_RESERVED_77,
APP_TIMER_RESERVED_78, APP_TIMER_RESERVED_79, APP_TIMER_RESERVED_80, CCG_ACTIVITY_TIMER_ID,
OTP_DEBOUNCE_TIMER_ID, TYPE_A_CUR_SENSE_TIMER_ID, TYPE_A_REG_SWITCH_TIMER_ID,
TYPE_A_PWM_STEP_TIMER_ID,
PB_DEBOUNCE_TIMER_ID, APP_PSOURCE_VBUS_SET_TIMER_ID, THROTTLE_TIMER_ID, THROTTLE_WAIT_FOR_POWER_TIMER_ID,
APP_BAD_SINK_TIMEOUT_TIMER, APP_FET_SOFT_START_TIMER_ID, APP_PPS_SNK_RECONTRACT_TIMER_ID,
APP_PSOURCE_SAFE_FET_ON_MONITOR_TIMER_ID,
APP_PSOURCE_SAFE_FET_ON_TIMER_ID, APP_TIMER_RESERVED_95, APP_POWER_SHARE_RECONTRACT_TIMER_ID,
CCG_LS_MASTER_PORT_DEBOUNCE_TIMER_ID,
CCG_LS_SLAVE_PORT_DEBOUNCE_TIMER_ID, CCG_LS_MASTER_WRITE_TIMER_ID, CCG_LS_HEART_BEAT_TIMER_ID,
USB_SUSPEND_TIMER,
USB_REMOTE_WAKEUP_TIMER, RIDGE_INIT_HPD_DEQUEUE_TIMER_ID, APP_TIMER_RESERVED_105,
APP_TIMER_RESERVED_106,
APP_TIMER_RESERVED_107, APP_TIMER_RESERVED_108, APP_TIMER_RESERVED_109, HPI_PD_CMD_TIMER,
I2C_SLAVE_SCB0_TIMER, I2C_SLAVE_SCB1_TIMER, I2C_SLAVE_SCB2_TIMER, I2C_SLAVE_SCB3_TIMER,
APP_TIMER_RESERVED_115, APP_TIMER_RESERVED_116, APP_TIMER_RESERVED_117, APP_TIMER_RESERVED_118,
APP_TIMER_RESERVED_119, APP_TIMER_RESERVED_120, APP_TIMER_RESERVED_121, APP_TIMER_RESERVED_122,
APP_TIMER_RESERVED_123, APP_TIMER_RESERVED_124, APP_TIMER_RESERVED_125, APP_TIMER_RESERVED_126,
APP_TIMER_RESERVED_127, APP_TIMER_RESERVED_128, APP_TIMER_RESERVED_129, APP_TIMER_RESERVED_130,
APP_TIMER_RESERVED_131, APP_TIMER_RESERVED_132, APP_TIMER_RESERVED_133, APP_TIMER_RESERVED_134,
APP_TIMER_RESERVED_135, APP_TIMER_RESERVED_136, APP_TIMER_RESERVED_137, APP_TIMER_RESERVED_138,
APP_TIMER_RESERVED_139, USER_TIMERS_START_ID = 140, APP_RETIMER_DISABLE_WAIT_TIMER
= 140, APP_AUTO_DR_SWAP_TIMER,
ICL_VSYS_STABLE_TIMER, TBT_MODE_EXIT_CHECK_TIMER, ICL_MUX_STATE_CHANGE_TIMER,
ICL_ADP_DETECT_DEB_TIMER,
BB_DBR_DEBUG_POLL_TIMER, BB_DBR_WAIT_TIMER, APP_HPD_DELAY_TIMER, APP_RETIMER_ENABLE_WAIT_TIMER,
APP_RETIMER_ENABLE_DELAY_TIMER, APP_RETIMER_DISABLE_DELAY_TIMER, ICL_RFU_EXIT_TIMER,
ICL_RETIMER_EXT_CONN_TIMER,
ICL_MUX_CONFIG_SS_TIMER, ICL_MUX_CONFIG_SAFE_TIMER, ICL_MUX_CONFIG RIDGE_CUSTOM_TIMER,
ICL_SOC_TIMEOUT_TIMER,
BB_DBR_DEBUG_WRITE_TIMER, ICL_ALT_MODE_EXIT_TIMER, ICL_HARD_RST_TIMER }

```

### 6.38.1 Detailed Description

Soft timer identifier definitions.

### 6.38.2 Enumeration Type Documentation

#### 6.38.2.1 ccg\_timer\_id\_t

```
enum ccg_timer_id_t
```

List of soft timer IDs defined for various applications. Values of the timer IDs should not be changed. Timer IDs from USER\_TIMERS\_START\_ID can be used by user code.

## Enumerator

|                                |                                                                                                       |
|--------------------------------|-------------------------------------------------------------------------------------------------------|
| PD_TIMERS_START_ID             | 000: Start index for USB-PD stack timers.                                                             |
| PD_CABLE_TIMER                 | 001: Timer used for cable capability check.                                                           |
| PD_NO_RESPONSE_TIMER           | 002: Response timer.                                                                                  |
| PD_CBL_DSC_ID_TIMER            | 003: Timer used for cable discovery state machine.                                                    |
| PD_CBL_DELAY_TIMER             | 004: Timer used to enforce cable delay.                                                               |
| PD_PHY_BUSY_TIMER              | 005: Timer used to handle PHY busy status.                                                            |
| PD_GOOD_CRC_TX_TIMER           | 006: GoodCRC timer.                                                                                   |
| PD_HARD_RESET_TX_TIMER         | 007: Hard reset transmit timer.                                                                       |
| PD_VCONN_SWAP_INITIATOR_TIMER  | 008: VConn swap initiator timer.                                                                      |
| PD_GENERIC_TIMER               | 009: Generic AMS timer.                                                                               |
| PD_PPS_TIMER                   | 010: PPS related timer.                                                                               |
| PD_SINK_TX_TIMER               | 011: PD 3.0 sink Rp flow control timer.                                                               |
| PD_DATA_RESET_COMP_TIMER       | 012: Reserved for future use.                                                                         |
| PD_TIMER_RESERVED_13           | 013: Reserved for future use.                                                                         |
| PD_TIMERS_END_ID               | 014: End index (inclusive) for USB-PD stack timers.                                                   |
| TYPE_C_TIMERS_START_ID         | 015: Start index for Type-C timers.                                                                   |
| TYPE_C_GENERIC_TIMER2          | 015: Generic Type-C state machine timer #2.                                                           |
| TYPE_C_GENERIC_TIMER1          | 016: Generic Type-C state machine timer #1.                                                           |
| TYPE_C_CC1_DEBOUNCE_TIMER      | 017: Timer used for CC1 debounce.                                                                     |
| TYPE_C_CC2_DEBOUNCE_TIMER      | 018: Timer used for CC2 debounce.                                                                     |
| TYPE_C_RD_DEBOUNCE_TIMER       | 019: Timer used for Rd debounce.                                                                      |
| TYPE_C_VBUS_DISCHARGE_TIMER    | 020: VBus discharge timer id.                                                                         |
| TYPE_C_ACTIVITY_TIMER          | 021: Type-C activity timer id.                                                                        |
| TYPE_C_RP_CHANGE_TIMER         | 022: Timer used to trigger current limit update after Rp change.                                      |
| TYPE_C_TIMER_RESERVED_23       | 023: Reserved for future use.                                                                         |
| TYPE_C_TIMERS_END_ID           | 024: End index (inclusive) for Type-C timers.                                                         |
| PD_OCP_DEBOUNCE_TIMER          | 025: Timer used for FW debounce of VBus OCP.                                                          |
| HPD_RX_ACTIVITY_TIMER_ID       | 026: Timer used for HPD receive handling.                                                             |
| PD_VCONN_OCP_DEBOUNCE_TIMER    | 027: Timer used for FW debounce of VConn OCP.                                                         |
| PD_TIMER_RESERVED_28           | 028: Reserved for future use.                                                                         |
| VBUS_DISCHARGE_SCHEDULE_TIMER  | 029: Timer used to monitor VBus discharge operation.                                                  |
| APP_TIMERS_START_ID            | 030: Start index for Application level timers.                                                        |
| APP_PSOURCE_EN_TIMER           | 030: Timer used to ensure timely completion of power source enable operation.                         |
| APP_PSOURCE_EN_MONITOR_TIMER   | 031: Timer used to monitor voltage during power source enable operation.                              |
| APP_PSOURCE_EN_HYS_TIMER       | 032: Timer used to add hysteresis at the end of a power source enable operation.                      |
| APP_PSOURCE_DIS_TIMER          | 033: Timer used to ensure timely completion of power source disable operation.                        |
| APP_PSOURCE_DIS_MONITOR_TIMER  | 034: Timer used to monitor voltage during power source disable operation.                             |
| APP_PSOURCE_CF_TIMER           | 035: Power source Current foldback restart timer ID.                                                  |
| APP_PSOURCE_DIS_EXT_DIS_TIMER  | 036: Timer used to discharge VBus for some extra time at the end of a power source disable operation. |
| APP_DB_SNK_FET_DIS_DELAY_TIMER | 037: Dead battery Sink Fet disable delay timer.                                                       |
| APP_PSINK_DIS_TIMER            | 038: Timer used to ensure timely completion of power sink disable operation.                          |

## Enumerator

|                                   |                                                                                                    |
|-----------------------------------|----------------------------------------------------------------------------------------------------|
| APP_PSINK_DIS_MONITOR_TIMER       | 039: Timer used to monitor voltage during power sink disable operation.                            |
| APP_VDM_BUSY_TIMER                | 040: Timer used to delay retry of VDMs due to BUSY responses or errors.                            |
| APP_AME_TIMEOUT_TIMER             | 041: Timer used to implement AME timeout.                                                          |
| APP_VBUS_OCP_OFF_TIMER            | 042: Timer used to disable VBus supply after OC fault.                                             |
| APP_VBUS_OVP_OFF_TIMER            | 043: Timer used to disable VBus supply after OV fault.                                             |
| APP_VBUS_UVP_OFF_TIMER            | 044: Timer used to disable VBus supply after UV fault.                                             |
| APP_VBUS_SCP_OFF_TIMER            | 045: Timer used to disable VBus supply after SC fault.                                             |
| APP_FAULT_RECOVERY_TIMER          | 046: App timer used to delay port enable after detecting a fault.                                  |
| APP_SBU_DELAYED_CONNECT_TIMER     | 047: Timer used to do delayed SBU connection in Thunderbolt mode.                                  |
| APP_MUX_DELAY_TIMER               | 048: Timer used to delay VDM response.                                                             |
| APP_MUX_POLL_TIMER                | 049: Timer used to MUX status.                                                                     |
| APP_CBL_DISC_TRIGGER_TIMER        | 050: Timer used to trigger cable discovery after a V5V supply change.                              |
| APP_V5V_CHANGE_DEBOUNCE_TIMER     | 051: Timer used to debounce V5V voltage changes.                                                   |
| APP_VCONN_RECOVERY_TIMER          | 052: Timer used to run Vconn swap after V5V was lost and recovered while UFP.                      |
| APP_OT_DETECTION_TIMER            | 053: Timer used to call OT measurement handler.                                                    |
| APP_PASC_VALLEY_TIMER_ID          | 054: Timer used to run PASC update task for valley algorithm.                                      |
| APP_PASC_TASK_TIMER_ID            | 055: Timer used to run PASC tasks periodically.                                                    |
| APP_CHUNKED_MSG_RESP_TIMER        | 056: Timer ID used to respond to chunked messages with NOT_SUPPORTED.                              |
| APP_RESET_VDM_LAYER_TIMER         | 057: Timer used to run reset of VDM layer.                                                         |
| APP_TIMER_RESERVED_58             | 058: Timer ID reserved for future use.                                                             |
| APP_TIMER_RESERVED_59             | 059: Timer ID reserved for future use.                                                             |
| APP_BB_ON_TIMER                   | 060: Timer used to provide delay between disabling the Billboard device and re-enabling it.        |
| APP_BB_OFF_TIMER                  | 061: Timer used to display USB billboard interface to save power.                                  |
| APP_TIMER_RESERVED_62             | 062: Timer ID reserved for future use.                                                             |
| APP_INITIATE_SWAP_TIMER           | 063: Timer used to initiate SWAP operations in DRP applications with a power/data role preference. |
| APP_INITIATE_SEND_IRQ_CLEAR_ACK   | 064: Timer used to initiate Virtual HPD IRQ CLEAR ACK to the Thunderbolt Controller.               |
| UCSI_CONNECT_EVENT_TIMER          | 065: UCSI Connect Event timer. This timer is used to Signal Connect event to OPM.                  |
| APP_VDM_NOT_SUPPORT_RESP_TIMER_ID | 066: VDM Not supported response timer.                                                             |
| UCSI_DPM_RETRY_TIMER              | 067: Timer ID used to retry DPM commands initiated by UCSI layer.                                  |
| APP_TIMER_RESERVED_68             | 068: Timer ID reserved for future use.                                                             |
| APP_TIMER_RESERVED_69             | 069: Timer ID reserved for future use.                                                             |
| APP_BC_TIMERS_START_ID            | 070: Start of Battery Charging State Machine timers.                                               |
| APP_BC_GENERIC_TIMER1             | 071: Generic timer #1 used by the BC state machine.                                                |
| APP_BC_GENERIC_TIMER2             | 072: Generic timer #2 used by the BC state machine.                                                |

## Enumerator

|                                          |                                                                               |
|------------------------------------------|-------------------------------------------------------------------------------|
| APP_BC_DP_DM_DEBOUNCE_TIMER              | 073: Timer used to debounce voltage changes on DP and DM pins.                |
| APP_BC_DETACH_DETECT_TIMER               | 074: Timer used to detect detach of a BC 1.2 sink while functioning as a CDP. |
| APP_CDP_DP_DM_POLL_TIMER                 | 075: Timer used to initiate DP/DM voltage polling while connected as a CDP.   |
| APP_TIMER_RESERVED_76                    | 076: Timer ID reserved for future use.                                        |
| APP_TIMER_RESERVED_77                    | 077: Timer ID reserved for future use.                                        |
| APP_TIMER_RESERVED_78                    | 078: Timer ID reserved for future use.                                        |
| APP_TIMER_RESERVED_79                    | 079: Timer ID reserved for future use.                                        |
| APP_TIMER_RESERVED_80                    | 080: Timer ID reserved for future use.                                        |
| CCG_ACTIVITY_TIMER_ID                    | 081: CCG activity timer ID.                                                   |
| OTP_DEBOUNCE_TIMER_ID                    | 082: OTP debounce timer ID.                                                   |
| TYPE_A_CUR_SENSE_TIMER_ID                | 083: TYPE-A current sense timer ID.                                           |
| TYPE_A_REG_SWITCH_TIMER_ID               | 084: TYPE-A regulator switch timer ID.                                        |
| TYPE_A_PWM_STEP_TIMER_ID                 | 085: TYPE-A port PWM step change timer.                                       |
| PB_DEBOUNCE_TIMER_ID                     | 086: Power Bank debounce timer ID.                                            |
| APP_PSOURCE_VBUS_SET_TIMER_ID            | 087: Power source VBUS set timer ID                                           |
| THROTTLE_TIMER_ID                        | 088: Power Throttling timer ID.                                               |
| THROTTLE_WAIT_FOR_PD_TIMER_ID            | 089: Power Throttling timer ID.                                               |
| APP_BAD_SINK_TIMEOUT_TIMER               | 090: PD bad sink timeout timer ID.                                            |
| APP_FET_SOFT_START_TIMER_ID              | 091: Timer used to control soft turn-on of power FET gate drivers.            |
| APP_PPS_SNK_RECONTRACT_TIMER_ID          | 092: Timer used by PPS sinks to send periodic request messages.               |
| APP_PSOURCE_SAFE_FET_ON_MONITOR_TIMER_ID | 093: Timer to monitor voltage during FET On operation.                        |
| APP_PSOURCE_SAFE_FET_ON_TIMER_ID         | 094: Timeout timer to set safe voltage during FET On operation.               |
| APP_TIMER_RESERVED_95                    | 095: Timer ID reserved for future use.                                        |
| APP_POWER_SHARE_RECONTRACT_TIMER_ID      | 096: Timer used to trigger recontract due to power allocation change.         |
| CCG_LS_MASTER_PORT_DEBOUNCE_TIMER_ID     | 097: Macro defines Master Debounce Timer ID.                                  |
| CCG_LS_SLAVE_PORT_DEBOUNCE_TIMER_ID      | 098: Macro defines Slave Debounce Timer ID.                                   |
| CCG_LS_MASTER_WRITE_TIMER_ID             | 099: Macro defines Master Write Timer ID.                                     |
| CCG_LS_HEART_BEAT_TIMER_ID               | 100: Macro defines Heart Beat Timer ID.                                       |
| USB_SUSPEND_TIMER                        | 101: This timer is used to check for USB bus suspend condition.               |
| USB_REMOTE_WAKEUP_TIMER                  | 102: This timer is used to signal remote wakeup.                              |
| RIDGE_INIT_HPD_DEQUEUE_TIMER_ID          | 103: Timer for initiating virtual HPD dequeue.                                |
| APP_TIMER_RESERVED_105                   | 105: Timer ID reserved for future use.                                        |
| APP_TIMER_RESERVED_106                   | 106: Timer ID reserved for future use.                                        |
| APP_TIMER_RESERVED_107                   | 107: Timer ID reserved for future use.                                        |
| APP_TIMER_RESERVED_108                   | 108: Timer ID reserved for future use.                                        |
| APP_TIMER_RESERVED_109                   | 109: Timer ID reserved for future use.                                        |
| HPI_PD_CMD_TIMER                         | 110: Timer ID used for HPI command retry checks.                              |
| I2C_SLAVE_SCB0_TIMER                     | 111: I2C transfer timeout for SCB0.                                           |
| I2C_SLAVE_SCB1_TIMER                     | 112: I2C transfer timeout for SCB1.                                           |
| I2C_SLAVE_SCB2_TIMER                     | 113: I2C transfer timeout for SCB2.                                           |
| I2C_SLAVE_SCB3_TIMER                     | 114: I2C transfer timeout for SCB3.                                           |

## Enumerator

|                                 |                                                                                                     |
|---------------------------------|-----------------------------------------------------------------------------------------------------|
| APP_TIMER_RESERVED_115          | 115: Timer ID reserved for future use.                                                              |
| APP_TIMER_RESERVED_116          | 116: Timer ID reserved for future use.                                                              |
| APP_TIMER_RESERVED_117          | 117: Timer ID reserved for future use.                                                              |
| APP_TIMER_RESERVED_118          | 118: Timer ID reserved for future use.                                                              |
| APP_TIMER_RESERVED_119          | 119: Timer ID reserved for future use.                                                              |
| APP_TIMER_RESERVED_120          | 120: Timer ID reserved for future use.                                                              |
| APP_TIMER_RESERVED_121          | 121: Timer ID reserved for future use.                                                              |
| APP_TIMER_RESERVED_122          | 122: Timer ID reserved for future use.                                                              |
| APP_TIMER_RESERVED_123          | 123: Timer ID reserved for future use.                                                              |
| APP_TIMER_RESERVED_124          | 124: Timer ID reserved for future use.                                                              |
| APP_TIMER_RESERVED_125          | 125: Timer ID reserved for future use.                                                              |
| APP_TIMER_RESERVED_126          | 126: Timer ID reserved for future use.                                                              |
| APP_TIMER_RESERVED_127          | 127: Timer ID reserved for future use.                                                              |
| APP_TIMER_RESERVED_128          | 128: Timer ID reserved for future use.                                                              |
| APP_TIMER_RESERVED_129          | 129: Timer ID reserved for future use.                                                              |
| APP_TIMER_RESERVED_130          | 130: Timer ID reserved for future use.                                                              |
| APP_TIMER_RESERVED_131          | 131: Timer ID reserved for future use.                                                              |
| APP_TIMER_RESERVED_132          | 132: Timer ID reserved for future use.                                                              |
| APP_TIMER_RESERVED_133          | 133: Timer ID reserved for future use.                                                              |
| APP_TIMER_RESERVED_134          | 134: Timer ID reserved for future use.                                                              |
| APP_TIMER_RESERVED_135          | 135: Timer ID reserved for future use.                                                              |
| APP_TIMER_RESERVED_136          | 136: Timer ID reserved for future use.                                                              |
| APP_TIMER_RESERVED_137          | 137: Timer ID reserved for future use.                                                              |
| APP_TIMER_RESERVED_138          | 138: Timer ID reserved for future use.                                                              |
| APP_TIMER_RESERVED_139          | 139: Timer ID reserved for future use.                                                              |
| USER_TIMERS_START_ID            | 140: Start of timer IDs left for generic solution level usage.                                      |
| APP_RETIMER_DISABLE_WAIT_TIMER  | 140: This timer is used to delay retimer disable.                                                   |
| APP_AUTO_DR_SWAP_TIMER          | 141: This timer is used to trigger DR_Swap after PD Contract.                                       |
| ICL_VSYS_STABLE_TIMER           | 142: This timer is let VSYS turn-on change to stabilize.                                            |
| TBT_MODE_EXIT_CHECK_TIMER       | 143: This timer is used to periodically check if TBT mode should be exited.                         |
| ICL_MUX_STATE_CHANGE_TIMER      | 144: This timer is used to check ICL mux state changes.                                             |
| ICL_ADP_DETECT_DEB_TIMER        | 145: This timer is used debounce barrel state change.                                               |
| BB_DBR_DEBUG_POLL_TIMER         | 146: This timer is used to retry reading the DEBUG register from the retimer.                       |
| BB_DBR_WAIT_TIMER               | 147: This timer is used to delay the retimer to start accepting the state change requests.          |
| APP_HPD_DELAY_TIMER             | 148: This timer is used to delay HPD events.                                                        |
| APP_RETIMER_ENABLE_WAIT_TIMER   | 149: This timer is used to delay retimer enable.                                                    |
| APP_RETIMER_ENABLE_DELAY_TIMER  | 150: This timer is used to drive the retimer pins to appropriate values as part of retimer enable.  |
| APP_RETIMER_DISABLE_DELAY_TIMER | 151: This timer is used to drive the retimer pins to appropriate values as part of retimer disable. |
| ICL_RFU_EXIT_TIMER              | 152: This timer is used to check Retimer FW Exit state changes.                                     |

## Enumerator

|                                   |                                                                             |
|-----------------------------------|-----------------------------------------------------------------------------|
| ICL_RETIMER_EXT_CONN_TIMER        | 153: This timer is used to check ICL Retimer extended connection update.    |
| ICL_MUX_CONFIG_SS_TIMER           | 154: This timer is used to check ICL mux config SS state changes.           |
| ICL_MUX_CONFIG_SAFE_TIMER         | 155: This timer is used to check ICL mux config Safe state changes.         |
| ICL_MUX_CONFIG_RIDGE_CUSTOM_TIMER | 156: This timer is used to check ICL mux config ridge custom state changes. |
| ICL_SOC_TIMEOUT_TIMER             | 157: Timer used to implement t_SOCAckTimeout.                               |
| BB_DBR_DEBUG_WRITE_TIMER          | 158: This timer is used to delay retimer debug mode register write.         |
| ICL_ALT_MODE_EXIT_TIMER           | 159: This timer is used for comm delay in the RFU entry sequence.           |
| ICL_HARD_RST_TIMER                | 160: This timer is used for comm delay during the Hard Reset Sequence.      |

## 6.39 system/utils.h File Reference

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

## Macros

- #define GET\_MAX(a, b) (((a) > (b)) ? (a) : (b))
- #define GET\_MIN(a, b) (((a) > (b)) ? (b) : (a))
- #define DIV\_ROUND\_UP(x, y) (((x) + ((y) - 1)) / (y))
- #define DIV\_ROUND\_NEAREST(x, y) (((x) + ((y) / 2)) / (y))
- #define MAKE\_WORD(hi, lo) (((uint16\_t)(hi) << 8) | ((uint16\_t)(lo)))
- #define MAKE\_DWORD(b3, b2, b1, b0)
- #define MAKE\_DWORD\_FROM\_WORD(hi, lo) (((uint32\_t)(hi) << 16) | ((uint32\_t)(lo)))
- #define WORD\_GET\_MSB(w) ((uint8\_t)((w) >> 8))
- #define WORD\_GET\_LSB(w) ((uint8\_t)((w) & 0xFF))
- #define BYTE\_GET\_UPPER\_NIBBLE(w) ((uint8\_t)((w) >> 4) & 0xF)
- #define BYTE\_GET\_LOWER\_NIBBLE(w) ((uint8\_t)((w) & 0xF))
- #define DWORD\_GET\_BYTE0(dw) ((uint8\_t)((dw) & 0xFF))
- #define DWORD\_GET\_BYTE1(dw) ((uint8\_t)((dw) >> 8) & 0xFF)
- #define DWORD\_GET\_BYTE2(dw) ((uint8\_t)((dw) >> 16) & 0xFF)
- #define DWORD\_GET\_BYTE3(dw) ((uint8\_t)((dw) >> 24) & 0xFF)
- #define MEM\_COPY(dest, src, size) memcpy((uint8\_t\*)(dest), (uint8\_t\*)(src), (size))
- #define MEM\_CMP(buf1, buf2, size) memcmp((uint8\_t\*)(buf1), (uint8\_t\*)(buf2), (size))
- #define MEM\_SET(dest, byte, size) memset((uint8\_t\*)(dest), (byte), (size))
- #define BUSY\_WAIT\_US(us) CyDelayCycles((uint32\_t)(us))
- #define REV\_BYTE\_ORDER(n, b, i)
- #define REG\_FIELD\_UPDATE(reg, field, value) (reg) = ((reg & ~(field##\_MASK)) | ((value) << field##\_POS))
- #define REG\_FIELD\_GET(reg, field) (((reg) & field##\_MASK) >> field##\_POS)



## Functions

- uint8\_t [mem\\_calculate\\_byte\\_checksum](#) (uint8\_t \*ptr, uint32\_t size)
- uint16\_t [mem\\_calculate\\_word\\_checksum](#) (uint16\_t \*ptr, uint32\_t size)
- uint32\_t [mem\\_calculate\\_dword\\_checksum](#) (uint32\_t \*ptr, uint32\_t size)
- uint16\_t [crc16](#) (uint16\_t crc, uint8\_t data)
- void [mem\\_copy\\_word](#) (uint32\_t \*dest, const uint32\_t \*source, uint32\_t size)
- void [mem\\_copy](#) (uint8\_t \*dest, const uint8\_t \*source, uint32\_t size)
- void [mem\\_set](#) (uint8\_t \*dest, uint8\_t c, uint32\_t size)
- uint32\_t [div\\_round\\_up](#) (uint32\_t x, uint32\_t y)
- int32\_t [apply\\_threshold](#) (int32\_t val, int8\_t percentage)
- void [event\\_group\\_set\\_event](#) (uint32\_t \*event\_map, uint8\_t event\_sel)
- void [event\\_group\\_set\\_events\\_by\\_val](#) (uint32\_t \*event\_map, uint32\_t event\_sel)
- void [event\\_group\\_clear\\_event](#) (uint32\_t \*event\_map, uint8\_t event\_sel)
- void [event\\_group\\_clear\\_events\\_by\\_val](#) (uint32\_t \*event\_map, uint32\_t event\_sel)
- uint8\_t [event\\_group\\_get\\_event](#) (volatile uint32\_t \*event\_map, bool clr\_stat)

### 6.39.1 Detailed Description

General utility macros and definitions for CCGx firmware stack.

### 6.39.2 Macro Definition Documentation

#### 6.39.2.1 BUSY\_WAIT\_US

```
#define BUSY_WAIT_US(
 us) CyDelayCycles ((uint32_t)(us))
```

Insert delay of the desired number of us using a busy spin-loop.

#### 6.39.2.2 BYTE\_GET\_LOWER\_NIBBLE

```
#define BYTE_GET_LOWER_NIBBLE(
 w) ((uint8_t)((w) & 0xF))
```

Retrieve the Lower nibble from a BYTE.

#### 6.39.2.3 BYTE\_GET\_UPPER\_NIBBLE

```
#define BYTE_GET_UPPER_NIBBLE(
 w) ((uint8_t)((w) >> 4) & 0xF)
```

Retrieve the Upper nibble from a BYTE.

#### 6.39.2.4 DIV\_ROUND\_NEAREST

```
#define DIV_ROUND_NEAREST(
 x,
 y) (((x) + ((y) / 2)) / (y))
```

Integer division - round to nearest integer.

### 6.39.2.5 DIV\_ROUND\_UP

```
#define DIV_ROUND_UP(
 x,
 y) ((x) + ((y) - 1)) / (y)
```

Integer division - round up to next integer.

### 6.39.2.6 DWORD\_GET\_BYTE0

```
#define DWORD_GET_BYTE0(
 dw) ((uint8_t)((dw) & 0xFF))
```

Retrieve the LSB from a DWORD.

### 6.39.2.7 DWORD\_GET\_BYTE1

```
#define DWORD_GET_BYTE1(
 dw) ((uint8_t)((dw) >> 8) & 0xFF)
```

Retrieve the bits 15-8 from a DWORD.

### 6.39.2.8 DWORD\_GET\_BYTE2

```
#define DWORD_GET_BYTE2(
 dw) ((uint8_t)((dw) >> 16) & 0xFF)
```

Retrieve the bits 23-16 from a DWORD.

### 6.39.2.9 DWORD\_GET\_BYTE3

```
#define DWORD_GET_BYTE3(
 dw) ((uint8_t)((dw) >> 24) & 0xFF)
```

Retrieve the MSB from a DWORD.

### 6.39.2.10 GET\_MAX

```
#define GET_MAX(
 a,
 b) ((a) > (b)) ? (a) : (b)
```

Get the maximum from among two numbers.

### 6.39.2.11 GET\_MIN

```
#define GET_MIN(
 a,
 b) ((a) > (b)) ? (b) : (a)
```

Get the minimum from among two numbers.

### 6.39.2.12 MAKE\_DWORD

```
#define MAKE_DWORD(
 b3,
```

```

 b2,
 b1,
 b0)

```

**Value:**

```

(((uint32_t)(b3) << 24) | ((uint32_t)(b2) << 16) | \
 ((uint32_t)(b1) << 8) | ((uint32_t)(b0)))

```

Combine four bytes to create one 32-bit DWORD.

**6.39.2.13 MAKE\_DWORD\_FROM\_WORD**

```

#define MAKE_DWORD_FROM_WORD(
 hi,
 lo) (((uint32_t)(hi) << 16) | ((uint32_t)(lo)))

```

Combine two WORDs to create one DWORD.

**6.39.2.14 MAKE\_WORD**

```

#define MAKE_WORD(
 hi,
 lo) (((uint16_t)(hi) << 8) | ((uint16_t)(lo)))

```

Combine two bytes to create one 16-bit WORD.

**6.39.2.15 MEM\_CMP**

```

#define MEM_CMP(
 buf1,
 buf2,
 size) memcmp ((uint8_t *) (buf1), (uint8_t *) (buf2), (size))

```

memcmp abstraction macro.

**6.39.2.16 MEM\_COPY**

```

#define MEM_COPY(
 dest,
 src,
 size) memcpy ((uint8_t *) (dest), (uint8_t *) (src), (size))

```

memcpy abstraction macro.

**6.39.2.17 MEM\_SET**

```

#define MEM_SET(
 dest,
 byte,
 size) memset ((uint8_t *) (dest), (byte), (size))

```

memset abstraction macro.

### 6.39.2.18 REG\_FIELD\_GET

```
#define REG_FIELD_GET(
 reg,
 field) ((reg) & field##_MASK) >> field##_POS
```

Retrieve a specific field from a register.

### 6.39.2.19 REG\_FIELD\_UPDATE

```
#define REG_FIELD_UPDATE(
 reg,
 field,
 value) (reg) = ((reg & ~(field##_MASK)) | ((value) << field##_POS))
```

Update a single field in a register. It first clears the field and then updates the data.

### 6.39.2.20 REV\_BYTE\_ORDER

```
#define REV_BYTE_ORDER(
 n,
 b,
 i)
```

**Value:**

```
(n) = ((uint32_t) (b)[(i)] << 24) \
 | ((uint32_t) (b)[(i) + 1] << 16) \
 | ((uint32_t) (b)[(i) + 2] << 8) \
 | ((uint32_t) (b)[(i) + 3]);
```

Macro to combine 4 bytes in reverse order to make a 32-bit integer.

**Parameters**

|          |                                      |
|----------|--------------------------------------|
| <i>n</i> | Final 32 bit number                  |
| <i>b</i> | Input array of bytes to be combined. |
| <i>i</i> | Index into the input byte array.     |

### 6.39.2.21 WORD\_GET\_LSB

```
#define WORD_GET_LSB(
 w) ((uint8_t)((w) & 0xFF))
```

Retrieve the LSB from a WORD.

### 6.39.2.22 WORD\_GET\_MSB

```
#define WORD_GET_MSB(
 w) ((uint8_t)((w) >> 8))
```

Retrieve the MSB from a WORD.

### 6.39.3 Function Documentation

#### 6.39.3.1 apply\_threshold()

```
int32_t apply_threshold (
 int32_t val,
 int8_t percentage)
```

Function to calculate threshold value based on input value and percentage which input value should be increased.

##### Parameters

|                   |                                                                         |
|-------------------|-------------------------------------------------------------------------|
| <i>val</i>        | Input value.                                                            |
| <i>percentage</i> | This value means how much (in percents) input value should be increased |

##### Returns

Sum of input value and calculated threshold

#### 6.39.3.2 crc16()

```
uint16_t crc16 (
 uint16_t crc,
 uint8_t data)
```

Function to calculate 16-bit CRC.

The function implements CRC-16 with polynomial  $x^{16} + x^{15} + x^2 + 1$  (0xA001).

##### Parameters

|             |                                              |
|-------------|----------------------------------------------|
| <i>crc</i>  | Original CRC value.                          |
| <i>data</i> | Data byte to be included in CRC computation. |

##### Returns

Updated CRC value including the new data byte.

#### 6.39.3.3 div\_round\_up()

```
uint32_t div_round_up (
 uint32_t x,
 uint32_t y)
```

Function to divide two unsigned integers and return the quotient rounded up to the nearest integer.

##### Parameters

|          |                                      |
|----------|--------------------------------------|
| <i>x</i> | The dividend value.                  |
| <i>y</i> | The divisor value. Must be non-zero. |

**Returns**

Quotient rounded up to nearest integer.

**6.39.3.4 event\_group\_clear\_event()**

```
void event_group_clear_event (
 uint32_t * event_map,
 uint8_t event_sel)
```

Function to clear a bit in an event group variable.

**Parameters**

|                  |                                                              |
|------------------|--------------------------------------------------------------|
| <i>event_map</i> | This holds the current status of the event group.            |
| <i>event_sel</i> | This specifies the position of the event/task to be cleared. |

**Returns**

None

**6.39.3.5 event\_group\_clear\_events\_by\_val()**

```
void event_group_clear_events_by_val (
 uint32_t * event_map,
 uint32_t event_sel)
```

Function to clear a bit in an event group variable.

**Parameters**

|                  |                                                           |
|------------------|-----------------------------------------------------------|
| <i>event_map</i> | This holds the current status of the event group.         |
| <i>event_sel</i> | This specifies one or more event/task bits to be cleared. |

**Returns**

None

**6.39.3.6 event\_group\_get\_event()**

```
uint8_t event_group_get_event (
 volatile uint32_t * event_map,
 bool clr_stat)
```

Get the next pending event/task from an event group variable.

**Parameters**

|                  |                                               |
|------------------|-----------------------------------------------|
| <i>event_map</i> | Original status of the event group.           |
| <i>clr_stat</i>  | Whether the event returned should be cleared. |

**Returns**

Position of the event/task which has been detected and needs to be processed.

**6.39.3.7 event\_group\_set\_event()**

```
void event_group_set_event (
 uint32_t * event_map,
 uint8_t event_sel)
```

Function to set a bit in an event group variable. Event groups are DWORD (uint32\_t) variables which hold a set of request bits which can be set/cleared independently. An event group facilitates management of a set of related tasks or status bits.

**Parameters**

|                  |                                                          |
|------------------|----------------------------------------------------------|
| <i>event_map</i> | This holds the current status of the event group.        |
| <i>event_sel</i> | This specifies the position of the event/task to be set. |

**Returns**

None

**6.39.3.8 event\_group\_set\_events\_by\_val()**

```
void event_group_set_events_by_val (
 uint32_t * event_map,
 uint32_t event_sel)
```

Function to set one or more task values in an event group. Event groups are DWORD (uint32\_t) variables which hold a set of request bits which can be set/cleared independently. An event group facilitates management of a set of related tasks or status bits.

**Parameters**

|                  |                                                              |
|------------------|--------------------------------------------------------------|
| <i>event_map</i> | This holds the current status of the event group.            |
| <i>event_sel</i> | This specifies a set of event/task bits which should be set. |

**Returns**

None

**6.39.3.9 mem\_calculate\_byte\_checksum()**

```
uint8_t mem_calculate_byte_checksum (
 uint8_t * ptr,
 uint32_t size)
```

Calculate the 2's complement binary checksum over a BYTE array.

This function calculates the checksum of the specified byte array. The checksum is a simple function calculated as the 2's complement of the binary sum of all bytes in the array. This checksum is used for the firmware binary as well as the configuration table.

#### Parameters

|             |                                     |
|-------------|-------------------------------------|
| <i>ptr</i>  | Pointer to the data array.          |
| <i>size</i> | Size of the array in BYTE elements. |

#### Returns

Checksum of the data array.

#### 6.39.3.10 mem\_calculate\_dword\_checksum()

```
uint32_t mem_calculate_dword_checksum (
 uint32_t * ptr,
 uint32_t size)
```

Calculate the 2's complement binary checksum over a DWORD array.

This function calculates the checksum of the specified DWORD array. The checksum is a simple function calculated as the 2's complement of the binary sum of all d-words in the array. This checksum is used for the firmware binary as well as the configuration table.

#### Parameters

|             |                                      |
|-------------|--------------------------------------|
| <i>ptr</i>  | Pointer to the data array.           |
| <i>size</i> | Size of the array in DWORD elements. |

#### Returns

Checksum of the data array.

#### 6.39.3.11 mem\_calculate\_word\_checksum()

```
uint16_t mem_calculate_word_checksum (
 uint16_t * ptr,
 uint32_t size)
```

Calculate the 2's complement binary checksum over a WORD array.

This function calculates the checksum of the specified WORD array. The checksum is a simple function calculated as the 2's complement of the binary sum of all words in the array.

#### Parameters

|             |                                     |
|-------------|-------------------------------------|
| <i>ptr</i>  | Pointer to the data array.          |
| <i>size</i> | Size of the array in WORD elements. |



**Returns**

Checksum of the data array.

**6.39.3.12 mem\_copy()**

```
void mem_copy (
 uint8_t * dest,
 const uint8_t * source,
 uint32_t size)
```

Function to do a byte-by-byte copy of data. Replacement for memcpy function. This function is used in cases where we need a memcpy equivalent that can be placed in a specific memory region (such as ROM).

**Parameters**

|               |                                       |
|---------------|---------------------------------------|
| <i>dest</i>   | Pointer to destination buffer.        |
| <i>source</i> | Pointer to source buffer.             |
| <i>size</i>   | Size of data to be copied (in bytes). |

**Returns**

None

**6.39.3.13 mem\_copy\_word()**

```
void mem_copy_word (
 uint32_t * dest,
 const uint32_t * source,
 uint32_t size)
```

Function to copy 32-bit data from one location to another.

**Parameters**

|               |                                     |
|---------------|-------------------------------------|
| <i>dest</i>   | Pointer to Destination.             |
| <i>source</i> | Pointer to source.                  |
| <i>size</i>   | Size of data in 32-bit DWORD units. |

**Returns**

None.

**6.39.3.14 mem\_set()**

```
void mem_set (
 uint8_t * dest,
 uint8_t c,
 uint32_t size)
```

Function to initialize a memory buffer. Replacement for memset function. This function is used in cases where we need a memset equivalent that can be placed in a specific memory region (such as ROM).

#### Parameters

|             |                                                     |
|-------------|-----------------------------------------------------|
| <i>dest</i> | Pointer to destination buffer.                      |
| <i>c</i>    | The data to be copied into each byte of the buffer. |
| <i>size</i> | Size of the buffer (in bytes).                      |

#### Returns

None

## 6.40 ucsi/ucsi.h File Reference

```
#include <stdbool.h>
#include <stdint.h>
#include <config.h>
#include "status.h"
#include "i2c.h"
#include "timer.h"
#include "ucsi_internal.h"
#include "pd.h"
#include "dpm.h"
```

#### Macros

- #define UCSI\_NOTIFICATION\_EN\_ALL (0xDBE7)
- #define UCSI\_NOTIFICATION\_EN\_REQ (0xDA05)
- #define UCSI\_BUFFER\_SIZE (HPI\_BUFFER\_SIZE)
- #define MAX\_DPM\_CMD\_RETRY\_COUNT (6u)
- #define CABLE\_CURR\_DFLT (18u)
- #define CABLE\_CURR\_3A (60u)
- #define CABLE\_CURR\_5A (100u)
- #define CABLE\_SPEED\_480MBPS (0x7820)
- #define CABLE\_SPEED\_5GBPS (0x0017)
- #define CABLE\_SPEED\_10GBPS (0x002B)
- #define CABLE\_VDO\_DIRECTION (0x0780)
- #define ALT\_MODES\_RECIPIENT\_CONNECTOR (0u)
- #define ALT\_MODES\_RECIPIENT\_SOP (1u)
- #define ALT\_MODES\_RECIPIENT\_SOP\_PRIME (2u)
- #define ALT\_MODES\_RECIPIENT\_SOP\_DPRIME (3u)
- #define VDM\_DISCOVER\_ID (1u)
- #define VDM\_DISCOVER\_SVID (2u)
- #define VDM\_DISCOVER\_MODES (3u)
- #define SRC\_CUR\_LEVEL\_DEF (0x00)
- #define SRC\_CUR\_LEVEL\_1\_5A (0x01)
- #define SRC\_CUR\_LEVEL\_3A (0x02)
- #define UCSI\_SET\_RP\_PPM\_DEFAULT (0)
- #define UCSI\_SET\_RP\_3A (1)
- #define UCSI\_SET\_RP\_1\_5A (2)
- #define UCSI\_SET\_RP\_DEF (3)

- #define `POM_NO_CONSUMER` (0x00)
- #define `POM_CUR_LEVEL_DEF` (0x01)
- #define `POM_BC` (0x02)
- #define `POM_PD` (0x03)
- #define `POM_CUR_LEVEL_1_5A` (0x04)
- #define `POM_CUR_LEVEL_3A` (0x05)
- #define `UCSI_READ_PENDING_EVENT` (7)
- #define `UCSI_READ_PENDING_MASK` (1 << `UCSI_READ_PENDING_EVENT`)

## Functions

- void `ucsi_init` (void)
- void `ucsi_task` (void)
- bool `is_port_connect_changed` (uint8\_t port)
- void `ucsi_pd_event_handler` (uint8\_t port, `app_evt_t` evt, const void \*data)
- void `ucsi_notify` (uint8\_t port, uint16\_t notification)
- void `ucsi_configure_send_vdm` (`sop_t` cmd\_sop, uint32\_t svid, uint32\_t cmd)
- bool `ucsi_sleep_allowed` (void)
- void `ucsi_refresh_conn_cnt` (void)

### 6.40.1 Detailed Description

USB Type-C Connector System Software Interface (UCSI) header file.

### 6.40.2 Macro Definition Documentation

#### 6.40.2.1 ALT\_MODES\_RECIPIENT\_CONNECTOR

```
#define ALT_MODES_RECIPIENT_CONNECTOR (0u)
```

Connector value for Alternate mode recipient.

#### 6.40.2.2 ALT\_MODES\_RECIPIENT\_SOP

```
#define ALT_MODES_RECIPIENT_SOP (1u)
```

SOP value for Alternate mode recipient.

#### 6.40.2.3 ALT\_MODES\_RECIPIENT\_SOP\_DPRIME

```
#define ALT_MODES_RECIPIENT_SOP_DPRIME (3u)
```

SOP" value for Alternate mode recipient.

#### 6.40.2.4 ALT\_MODES\_RECIPIENT\_SOP\_PRIME

```
#define ALT_MODES_RECIPIENT_SOP_PRIME (2u)
```

SOP' value for Alternate mode recipient.

#### 6.40.2.5 CABLE\_CURR\_3A

```
#define CABLE_CURR_3A (60u)
```

Type-C cable current capacity of 3A represented in 50 mA units.

#### 6.40.2.6 CABLE\_CURR\_5A

```
#define CABLE_CURR_5A (100u)
```

Type-C cable current capacity of 5A represented in 50 mA units.

#### 6.40.2.7 CABLE\_CURR\_DFLT

```
#define CABLE_CURR_DFLT (18u)
```

Type-C default current capacity represented in 50 mA units.

#### 6.40.2.8 CABLE\_SPEED\_10GBPS

```
#define CABLE_SPEED_10GBPS (0x002B)
```

USB 3.1 Type-C cable data speed capability (10 Gbps).

#### 6.40.2.9 CABLE\_SPEED\_480MBPS

```
#define CABLE_SPEED_480MBPS (0x7820)
```

USB 2.0 Type-C cable data speed capability (480 Mbps)

#### 6.40.2.10 CABLE\_SPEED\_5GBPS

```
#define CABLE_SPEED_5GBPS (0x0017)
```

USB 3.0 Type-C cable data speed capability (5 Gbps).

#### 6.40.2.11 CABLE\_VDO\_DIRECTION

```
#define CABLE_VDO_DIRECTION (0x0780)
```

Mask for directionality flags in USB-PD Cable VDO.

#### 6.40.2.12 MAX\_DPM\_CMD\_RETRY\_COUNT

```
#define MAX_DPM_CMD_RETRY_COUNT (6u)
```

Maximum retry count for DPM commands.

#### 6.40.2.13 POM\_BC

```
#define POM_BC (0x02)
```

Power Operation Mode (POM) of connector: BC 1.2

**6.40.2.14 POM\_CUR\_LEVEL\_1\_5A**

```
#define POM_CUR_LEVEL_1_5A (0x04)
```

Power Operation Mode (POM) of connector: Type-C 1.5 A.

**6.40.2.15 POM\_CUR\_LEVEL\_3A**

```
#define POM_CUR_LEVEL_3A (0x05)
```

Power Operation Mode (POM) of connector: Type-C 3 A.

**6.40.2.16 POM\_CUR\_LEVEL\_DEF**

```
#define POM_CUR_LEVEL_DEF (0x01)
```

Power Operation Mode (POM) of connector: Type-C default.

**6.40.2.17 POM\_NO\_CONSUMER**

```
#define POM_NO_CONSUMER (0x00)
```

Power Operation Mode (POM) of connector: reserved.

**6.40.2.18 POM\_PD**

```
#define POM_PD (0x03)
```

Power Operation Mode (POM) of connector: USB-PD.

**6.40.2.19 SRC\_CUR\_LEVEL\_1\_5A**

```
#define SRC_CUR_LEVEL_1_5A (0x01)
```

Current source value: 1.5 A

**6.40.2.20 SRC\_CUR\_LEVEL\_3A**

```
#define SRC_CUR_LEVEL_3A (0x02)
```

Current source value: 3 A

**6.40.2.21 SRC\_CUR\_LEVEL\_DEF**

```
#define SRC_CUR_LEVEL_DEF (0x00)
```

Current source value: Type-C default (900 mA).

**6.40.2.22 UCSI\_BUFFER\_SIZE**

```
#define UCSI_BUFFER_SIZE (HPI_BUFFER_SIZE)
```

Size of data receive buffer to be used for the UCSI interface. This size includes two bytes of register address and 256 bytes corresponding to the maximum write size.

#### 6.40.2.23 UCSI\_NOTIFICATION\_EN\_ALL

```
#define UCSI_NOTIFICATION_EN_ALL (0xDBE7)
```

Value to enable all (required & optional) UCSI Notification.

#### 6.40.2.24 UCSI\_NOTIFICATION\_EN\_REQ

```
#define UCSI_NOTIFICATION_EN_REQ (0xDA05)
```

Value to enable only Required UCSI Notification.

#### 6.40.2.25 UCSI\_READ\_PENDING\_EVENT

```
#define UCSI_READ_PENDING_EVENT (7)
```

UCSI Read Pending status bit in INTERRUPT register. Note: In the AMD TED platform, Bit 3 of the INTERRUPT register is used to indicate that a UCSI read is pending. This is changed to bit 7 in the final spec that was released to other customers. Change as necessary.

#### 6.40.2.26 UCSI\_READ\_PENDING\_MASK

```
#define UCSI_READ_PENDING_MASK (1 << UCSI_READ_PENDING_EVENT)
```

Mask to be applied for UCSI Read Pending Event.

#### 6.40.2.27 UCSI\_SET\_RP\_1\_5A

```
#define UCSI_SET_RP_1_5A (2)
```

UCSI command to set PPM power level to 1.5 A.

#### 6.40.2.28 UCSI\_SET\_RP\_3A

```
#define UCSI_SET_RP_3A (1)
```

UCSI command to set PPM power level to 3 A.

#### 6.40.2.29 UCSI\_SET\_RP\_DEF

```
#define UCSI_SET_RP_DEF (3)
```

UCSI command to set PPM power level to Type-C default.

#### 6.40.2.30 UCSI\_SET\_RP\_PPM\_DEFAULT

```
#define UCSI_SET_RP_PPM_DEFAULT (0)
```

UCSI command to set PPM power level to default value.

#### 6.40.2.31 VDM\_DISCOVER\_ID

```
#define VDM_DISCOVER_ID (1u)
```

VDM command value for Discover Identity.

### 6.40.2.32 VDM\_DISCOVER\_MODES

```
#define VDM_DISCOVER_MODES (3u)
```

VDM command value for Discover Modes

### 6.40.2.33 VDM\_DISCOVER\_SVID

```
#define VDM_DISCOVER_SVID (2u)
```

VDM command value for Discover SVID

## 6.40.3 Function Documentation

### 6.40.3.1 is\_port\_connect\_changed()

```
bool is_port_connect_changed (
 uint8_t port)
```

Check any change in port connection.

#### Parameters

|             |            |
|-------------|------------|
| <i>port</i> | port index |
|-------------|------------|

#### Returns

true if the port connect is changed, false otherwise.

### 6.40.3.2 ucsi\_configure\_send\_vdm()

```
void ucsi_configure_send_vdm (
 sop_t cmd_sop,
 uint32_t svid,
 uint32_t cmd)
```

Function to configure VDM for getting Alt Mode SVIDs & Modes.

#### Parameters

|                |                                         |
|----------------|-----------------------------------------|
| <i>cmd_sop</i> | Enum of the SOP (Start Of Frame) types. |
| <i>svid</i>    | Standard ID / Vendor ID                 |
| <i>cmd</i>     | DPM command                             |

#### Returns

None

**6.40.3.3 ucsi\_init()**

```
void ucsi_init (
 void)
```

Initialize the UCSI interface.

**Returns**

None

**6.40.3.4 ucsi\_notify()**

```
void ucsi_notify (
 uint8_t port,
 uint16_t notification)
```

Function to update UCSI notifications.

**Parameters**

|                     |                               |
|---------------------|-------------------------------|
| <i>port</i>         | port index                    |
| <i>notification</i> | UCSI notification to be sent. |

**Returns**

None

**6.40.3.5 ucsi\_pd\_event\_handler()**

```
void ucsi_pd_event_handler (
 uint8_t port,
 app_evt_t evt,
 const void * data)
```

Handler for PD events reported from the stack. Internal function used to receive PD events from the stack and to update the UCSI registers.

**Parameters**

|             |                                                                                                               |
|-------------|---------------------------------------------------------------------------------------------------------------|
| <i>port</i> | PD port corresponding to the event.                                                                           |
| <i>evt</i>  | Event that is being notified.                                                                                 |
| <i>data</i> | Data associated with the event. This is an opaque pointer that needs to be de-referenced based on event type. |

**Returns**

None



#### 6.40.3.6 ucsi\_refresh\_conn\_cnt()

```
void ucsi_refresh_conn_cnt (
 void)
```

Update the number of connectors on the system.

##### Returns

None

#### 6.40.3.7 ucsi\_sleep\_allowed()

```
bool ucsi_sleep_allowed (
 void)
```

Check if the UCSI module can be put to sleep.

##### Returns

true if the UCSI module can be put to sleep, false otherwise.

#### 6.40.3.8 ucsi\_task()

```
void ucsi_task (
 void)
```

UCSI task handler.

This function handles the commands from the EC through the UCSI registers. UCSI writes from the EC are handled in interrupt context, all UCSI commands are sent in sequence by OPM. There is no event queue maintained for UCSI interface. Instead, CCGx maintains a bit map of pending events for each port. CCGx forwards one connector change at a time to OPM. OPM can retrieve the connector status and receive a bit-map of events. The `ucsi_task` is expected to be called periodically from the main task loop of the firmware application.

##### Returns

None



# Index

ADC\_VBUS\_MIN\_OVP\_LEVEL  
app.h, 235

ADP\_EDGE\_NONE  
icl.h, 286

ADP\_NEGEDGE  
icl.h, 286

ADP\_POSEDGE  
icl.h, 286

ADP\_STATE\_INIT  
icl.h, 286

ALT\_MODE\_EVT\_DATA\_IDX  
alt\_modes\_mngr.h, 203

ALT\_MODE\_EVT\_IDX  
alt\_modes\_mngr.h, 203

ALT\_MODE\_EVT\_SIZE  
alt\_modes\_mngr.h, 204

ALT\_MODES\_RECIPIENT\_CONNECTOR  
ucsi.h, 577

ALT\_MODES\_RECIPIENT\_SOP\_DPRIME  
ucsi.h, 577

ALT\_MODES\_RECIPIENT\_SOP\_PRIME  
ucsi.h, 577

ALT\_MODES\_RECIPIENT\_SOP  
ucsi.h, 577

AM\_SVID\_CONFIG\_OFFSET\_IDX  
alt\_modes\_mngr.h, 204

AM\_SVID\_CONFIG\_SIZE\_IDX  
alt\_modes\_mngr.h, 204

APP\_AME\_TIMEOUT\_TIMER\_PERIOD  
app.h, 235

APP\_AUTO\_DR\_SWAP\_TRY\_PERIOD  
app.h, 235

APP\_BAD\_SINK\_TIMEOUT\_TIMER\_PERIOD  
app.h, 235

APP\_BB\_ON\_TIMER\_PERIOD  
app.h, 236

APP\_BC\_AFC\_DETECT\_TIMER\_PERIOD  
app.h, 236

APP\_BC\_APPLE\_DETECT\_TIMER\_PERIOD  
app.h, 236

APP\_BC\_CDP\_SM\_TIMER\_PERIOD  
app.h, 236

APP\_BC\_DCP\_DETECT\_TIMER\_PERIOD  
app.h, 236

APP\_BC\_DP\_DM\_DEBOUNCE\_TIMER\_PERIOD  
app.h, 236

APP\_BC\_GLITCH\_BC\_DONE\_TIMER\_PERIOD  
app.h, 236

APP\_BC\_GLITCH\_DM\_HIGH\_TIMER\_PERIOD  
app.h, 236

APP\_BC\_SINK\_CONTACT\_STABLE\_TIMER\_PERIOD  
app.h, 237

APP\_BC\_V\_NEW\_REQUEST\_TIMER\_PERIOD  
app.h, 237

APP\_BC\_VBUS\_CYCLE\_TIMER\_PERIOD  
app.h, 237

APP\_BC\_VDMSRC\_EN\_DIS\_PERIOD  
app.h, 237

APP\_BC\_VDP\_DM\_SRC\_ON\_PERIOD  
app.h, 237

APP\_CABLE\_POWER\_UP\_DELAY  
app.h, 237

APP\_CABLE\_VDM\_START\_DELAY  
app.h, 237

APP\_CBL\_DISC\_TIMER\_PERIOD  
app.h, 237

APP\_DB\_SNK\_FET\_DIS\_DELAY\_TIMER\_PERIOD  
app.h, 237

APP\_DR\_SWAP\_PENDING  
app.h, 238

APP\_FAULT\_RECOVERY\_MAX\_WAIT  
app.h, 238

APP\_FAULT\_RECOVERY\_TIMER\_PERIOD  
app.h, 238

APP\_INITIATE\_DR\_SWAP\_TIMER\_PERIOD  
app.h, 238

APP\_INITIATE\_PR\_SWAP\_TIMER\_PERIOD  
app.h, 238

APP\_INITIATE\_SEND\_IRQ\_CLEAR\_ACK\_PERIOD  
app.h, 238

APP\_MAX\_SWAP\_ATTEMPT\_COUNT  
app.h, 238

APP\_OT\_DETECTION\_TIMER\_PERIOD  
app.h, 238

APP\_OT\_HIGH\_TEMP  
app.h, 238

APP\_OT\_LOW\_TEMP  
app.h, 238

APP\_OT\_ROOM\_TEMP  
app.h, 239

APP\_OT\_VBE\_25\_C\_TEMP\_ADDR  
app.h, 239

APP\_OT\_VBE\_HIGH\_TEMP\_ADDR  
app.h, 239

APP\_OT\_VBE\_LOW\_TEMP\_ADDR  
app.h, 239

APP\_PB\_VBATT\_DEBOUNCE\_IN\_MS  
app.h, 239

APP\_PPS\_SNK\_CONTRACT\_PERIOD  
     pdo.h, 302  
 APP\_PPS\_SNK\_CONTRACT\_RETRY\_PERIOD  
     pdo.h, 302  
 APP\_PR\_SWAP\_PENDING  
     app.h, 239  
 APP\_PSINK\_DIS\_MONITOR\_TIMER\_PERIOD  
     app.h, 239  
 APP\_PSINK\_DIS\_TIMER\_PERIOD  
     app.h, 239  
 APP\_PSINK\_DIS\_VBUS\_IN\_DIS\_PERIOD  
     app.h, 239  
 APP\_PSOURCE\_CF\_TIMER\_PERIOD  
     app.h, 240  
 APP\_PSOURCE\_DIS\_EXT\_DIS\_TIMER\_PERIOD  
     app.h, 240  
 APP\_PSOURCE\_SAFE\_FET\_ON\_MONITOR\_TIMER←  
     R\_PERIOD  
     app.h, 240  
 APP\_PSOURCE\_VBUS\_SET\_TIMER\_PERIOD  
     app.h, 240  
 APP\_RESET\_VDM\_TIMER\_PERIOD  
     app.h, 240  
 APP\_RETIMER\_DISABLE\_DELAY  
     app.h, 240  
 APP\_RETIMER\_ENABLE\_DELAY  
     app.h, 240  
 APP\_SBU\_DELAYED\_CONNECT\_PERIOD  
     app.h, 240  
 APP\_UFP\_RECOV\_VCONN\_SWAP\_TIMER\_PERIOD  
     app.h, 240  
 APP\_VCONN\_RECOVERY\_PERIOD  
     app.h, 241  
 APP\_VCONN\_SWAP\_PENDING  
     app.h, 241  
 APP\_VDM\_BUSY\_TIMER\_PERIOD  
     app.h, 241  
 APP\_VDM\_FAIL\_RETRY\_PERIOD  
     app.h, 241  
 APPLE\_AMP\_1A  
     battery\_charging.h, 264  
 APPLE\_AMP\_2\_1A  
     battery\_charging.h, 264  
 APPLE\_AMP\_2\_4A  
     battery\_charging.h, 265  
 ATCH\_TGT  
     alt\_modes\_mgr.h, 204  
 AUTO\_CTRL\_MESSAGE\_GOODCRC\_MASK\_CFG  
     pdss\_hal.h, 444  
 AUTO\_DATA\_MESSAGE\_GOODCRC\_MASK\_CFG  
     pdss\_hal.h, 444  
 AUTO\_EXTD\_MESSAGE\_GOODCRC\_MASK\_CFG  
     pdss\_hal.h, 444  
 acc\_mode\_disable  
     custom\_host\_cfg\_settings\_t, 67  
 act\_cbl\_vdo  
     pd\_do\_t, 113  
 act\_cbl\_vdo1  
     pd\_do\_t, 113  
 act\_cbl\_vdo2  
     pd\_do\_t, 113  
 active\_boot\_app  
     sys\_fw\_metadata\_t, 179  
 active\_el  
     pd\_do\_t::ACT\_CBL\_VDO\_2, 26  
 actv\_swap\_count  
     app\_status\_t, 43  
 actv\_swap\_delay  
     app\_status\_t, 43  
 actv\_swap\_type  
     app\_status\_t, 44  
 adapter  
     pd\_do\_t::TBT\_UFP\_VDO, 182  
     pd\_do\_t::TBT\_VDO, 183  
 ado\_alert  
     pd\_do\_t, 113  
 afc\_src\_cap\_cnt  
     chg\_cfg\_params\_t, 63  
 afc\_src\_caps  
     chg\_cfg\_params\_t, 63  
 afc\_src\_cur\_match\_byte  
     bc\_status\_t, 58  
 afc\_src\_is\_matched  
     bc\_status\_t, 58  
 afc\_src\_last\_match\_byte  
     bc\_status\_t, 58  
 afc\_src\_match\_count  
     bc\_status\_t, 59  
 afc\_src\_matched\_byte  
     bc\_status\_t, 59  
 afc\_src\_msg\_count  
     bc\_status\_t, 59  
 afc\_tx\_active  
     bc\_status\_t, 59  
 alert  
     dpm\_status\_t, 75  
 alt\_cfg\_settings\_t, 29  
     dfp\_mask, 30  
     table\_len, 30  
     ufp\_mask, 30  
 alt\_mode  
     alt\_mode\_evt\_t::ALT\_MODE\_EVT, 30  
 alt\_mode\_app\_cbk\_t  
     alt\_modes\_mgr.h, 206  
 alt\_mode\_app\_cmd\_t  
     alt\_modes\_mgr.h, 207  
 alt\_mode\_app\_evt\_t  
     alt\_modes\_mgr.h, 207  
 alt\_mode\_cbk\_t  
     alt\_modes\_mgr.h, 207  
 alt\_mode\_entered  
     app\_status\_t, 44  
 alt\_mode\_event  
     alt\_mode\_evt\_t, 32  
 alt\_mode\_event\_data  
     alt\_mode\_evt\_t, 32

- alt\_mode\_evt
  - alt\_mode\_evt\_t::ALT\_MODE\_EVT, 31
- alt\_mode\_evt\_t, 32
  - alt\_mode\_event, 32
  - alt\_mode\_event\_data, 32
  - val, 32
- alt\_mode\_evt\_t::ALT\_MODE\_EVT\_DATA, 31
  - evt\_data, 31
  - evt\_type, 31
- alt\_mode\_evt\_t::ALT\_MODE\_EVT, 30
  - alt\_mode, 30
  - alt\_mode\_evt, 31
  - data\_role, 31
  - svid, 31
- alt\_mode\_get\_status
  - alt\_modes\_mngr.h, 209
- alt\_mode\_hw.h
  - alt\_mode\_hw\_deinit, 197
  - alt\_mode\_hw\_is\_idle, 198
  - alt\_mode\_hw\_set\_cbk, 198
  - alt\_mode\_hw\_sleep, 198
  - alt\_mode\_hw\_t, 196
  - alt\_mode\_hw\_wakeup, 199
  - dp\_snk\_get\_hpd\_state, 199
  - eval\_app\_alt\_hw\_cmd, 199
  - eval\_hpd\_cmd, 200
  - eval\_mux\_cmd, 200
  - get\_mux\_state, 200
  - HPD\_DISABLE\_CMD, 196
  - HPD\_ENABLE\_CMD, 196
  - ignore\_mux\_changes, 201
  - mux\_poll\_status\_t, 196
  - mux\_select\_t, 197
  - NO\_DATA, 196
  - set\_mux, 201
- alt\_mode\_hw\_deinit
  - alt\_mode\_hw.h, 197
- alt\_mode\_hw\_evt\_t, 33
  - hw\_evt, 34
  - val, 34
- alt\_mode\_hw\_evt\_t::ALT\_MODE\_HW\_EVT, 33
  - data\_role, 33
  - evt\_data, 33
  - hw\_type, 33
- alt\_mode\_hw\_is\_idle
  - alt\_mode\_hw.h, 198
- alt\_mode\_hw\_set\_cbk
  - alt\_mode\_hw.h, 198
- alt\_mode\_hw\_sleep
  - alt\_mode\_hw.h, 198
- alt\_mode\_hw\_t
  - alt\_mode\_hw.h, 196
- alt\_mode\_hw\_wakeup
  - alt\_mode\_hw.h, 199
- alt\_mode\_id
  - alt\_mode\_info\_t, 35
  - alt\_mode\_reg\_info\_t, 37
- alt\_mode\_info\_t, 34
  - alt\_mode\_id, 35
  - app\_evt\_data, 35
  - app\_evt\_needed, 35
  - cbk, 35
  - cbl\_obj\_pos, 35
  - custom\_att\_obj\_pos, 35
  - eval\_app\_cmd, 35
  - is\_active, 35
  - mode\_state, 35
  - obj\_pos, 36
  - set\_mux\_isolate, 36
  - sop\_state, 36
  - uvdm\_supp, 36
  - vdm\_header, 36
  - vdo, 36
  - vdo\_max\_numb, 36
  - vdo\_numb, 36
- alt\_mode\_layer\_reset
  - alt\_modes\_mngr.h, 209
- alt\_mode\_mngr\_exit\_all
  - alt\_modes\_mngr.h, 209
- alt\_mode\_mngr\_reset\_info
  - alt\_modes\_mngr.h, 210
- alt\_mode\_mngr\_sleep
  - alt\_modes\_mngr.h, 210
- alt\_mode\_mngr\_state\_t
  - alt\_modes\_mngr.h, 208
- alt\_mode\_mngr\_wakeup
  - alt\_modes\_mngr.h, 210
- alt\_mode\_reg\_info\_t, 37
  - alt\_mode\_id, 37
  - app\_evt, 37
  - atch\_tgt\_info, 37
  - atch\_type, 37
  - cbl\_sop\_flag, 37
  - data\_role, 37
  - svid\_emca\_vdo, 38
  - svid\_vdo, 38
- alt\_mode\_state\_t
  - alt\_modes\_mngr.h, 208
- alt\_mode\_tbl\_offset
  - pd\_port\_config\_t, 125
- alt\_mode\_trig\_mask
  - app\_status\_t, 44
- alt\_mode\_trigger
  - pd\_port\_config\_t, 125
- alt\_modes
  - pd\_do\_t::UFP\_VDO\_1, 186
- alt\_modes\_mngr.h
  - ALT\_MODE\_EVT\_DATA\_IDX, 203
  - ALT\_MODE\_EVT\_IDX, 203
  - ALT\_MODE\_EVT\_SIZE, 204
  - AM\_SVID\_CONFIG\_OFFSET\_IDX, 204
  - AM\_SVID\_CONFIG\_SIZE\_IDX, 204
  - ATCH\_TGT, 204
  - alt\_mode\_app\_cbk\_t, 206
  - alt\_mode\_app\_cmd\_t, 207
  - alt\_mode\_app\_evt\_t, 207

- alt\_mode\_cbk\_t, 207
- alt\_mode\_get\_status, 209
- alt\_mode\_layer\_reset, 209
- alt\_mode\_mgr\_exit\_all, 209
- alt\_mode\_mgr\_reset\_info, 210
- alt\_mode\_mgr\_sleep, 210
- alt\_mode\_mgr\_state\_t, 208
- alt\_mode\_mgr\_wakeup, 210
- alt\_mode\_state\_t, 208
- CABLE, 204
- CBL\_DIR\_SUPP\_MASK, 204
- DFP\_ALT\_MODE\_HPI\_OFFSET, 204
- EMPTY\_VDO, 204
- EN\_FLAG\_MASK, 204
- EXIT\_ALL\_MODES, 205
- eval\_app\_alt\_mode\_cmd, 211
- eval\_rec\_vdm, 211
- FULL\_MASK, 205
- fail\_status\_t, 208
- form\_alt\_mode\_event, 211
- get\_alt\_mode\_numb, 212
- get\_alt\_modes\_config\_svid\_idx, 212
- get\_custom\_svid, 213
- get\_mode\_info, 213
- get\_svid\_from\_idx, 213
- get\_vdm\_buff, 214
- IS\_FLAG\_CHECKED, 205
- is\_alt\_mode\_mgr\_idle, 214
- is\_svid\_supported, 214
- MAX\_RETRY\_CNT, 205
- MAX\_SUPP\_ALT\_MODES, 205
- MODE\_NOT\_SUPPORTED, 205
- NO\_DATA, 205
- NONE\_MODE\_MASK, 205
- NONE\_VDO, 205
- REMOVE\_FLAG, 206
- reg\_alt\_mode\_mgr, 215
- reset\_alt\_mode\_info, 215
- SET\_FLAG, 206
- set\_alt\_mode\_mask, 215
- set\_custom\_svid, 216
- UFP\_ALT\_MODE\_HPI\_MASK, 206
- VDM\_HDR, 206
- VDO\_START\_IDX, 206
- vdm\_task\_mgr\_alt\_mode\_process, 216
- ama\_fw\_ver
  - pd\_do\_t::STD\_AMA\_VDO\_PD3, 169
  - pd\_do\_t::STD\_AMA\_VDO, 167
- ama\_hw\_ver
  - pd\_do\_t::STD\_AMA\_VDO\_PD3, 169
  - pd\_do\_t::STD\_AMA\_VDO, 167
- ama\_vdo
  - atch\_tgt\_info\_t, 49
- apdo\_t
  - pd.h, 384
- apdo\_type
  - pd\_do\_t::PPS\_SNK, 143
  - pd\_do\_t::PPS\_SRC, 145
- app.h
  - ADC\_VBUS\_MIN\_OVP\_LEVEL, 235
  - APP\_AME\_TIMEOUT\_TIMER\_PERIOD, 235
  - APP\_AUTO\_DR\_SWAP\_TRY\_PERIOD, 235
  - APP\_BAD\_SINK\_TIMEOUT\_TIMER\_PERIOD, 235
  - APP\_BB\_ON\_TIMER\_PERIOD, 236
  - APP\_BC\_AFC\_DETECT\_TIMER\_PERIOD, 236
  - APP\_BC\_APPLE\_DETECT\_TIMER\_PERIOD, 236
  - APP\_BC\_CDP\_SM\_TIMER\_PERIOD, 236
  - APP\_BC\_DCP\_DETECT\_TIMER\_PERIOD, 236
  - APP\_BC\_DP\_DM\_DEBOUNCE\_TIMER\_PERIOD, 236
  - APP\_BC\_GLITCH\_BC\_DONE\_TIMER\_PERIOD, 236
  - APP\_BC\_GLITCH\_DM\_HIGH\_TIMER\_PERIOD, 236
  - APP\_BC\_SINK\_CONTACT\_STABLE\_TIMER\_PERIOD, 237
  - APP\_BC\_V\_NEW\_REQUEST\_TIMER\_PERIOD, 237
  - APP\_BC\_VBUS\_CYCLE\_TIMER\_PERIOD, 237
  - APP\_BC\_VDMSRC\_EN\_DIS\_PERIOD, 237
  - APP\_BC\_VDP\_DM\_SRC\_ON\_PERIOD, 237
  - APP\_CABLE\_POWER\_UP\_DELAY, 237
  - APP\_CABLE\_VDM\_START\_DELAY, 237
  - APP\_CBL\_DISC\_TIMER\_PERIOD, 237
  - APP\_DB\_SNK\_FET\_DIS\_DELAY\_TIMER\_PERIOD, 237
  - APP\_DR\_SWAP\_PENDING, 238
  - APP\_FAULT\_RECOVERY\_MAX\_WAIT, 238
  - APP\_FAULT\_RECOVERY\_TIMER\_PERIOD, 238
  - APP\_INITIATE\_DR\_SWAP\_TIMER\_PERIOD, 238
  - APP\_INITIATE\_PR\_SWAP\_TIMER\_PERIOD, 238
  - APP\_INITIATE\_SEND\_IRQ\_CLEAR\_ACK\_TIMER\_PERIOD, 238
  - APP\_MAX\_SWAP\_ATTEMPT\_COUNT, 238
  - APP\_OT\_DETECTION\_TIMER\_PERIOD, 238
  - APP\_OT\_HIGH\_TEMP, 238
  - APP\_OT\_LOW\_TEMP, 238
  - APP\_OT\_ROOM\_TEMP, 239
  - APP\_OT\_VBE\_25\_C\_TEMP\_ADDR, 239
  - APP\_OT\_VBE\_HIGH\_TEMP\_ADDR, 239
  - APP\_OT\_VBE\_LOW\_TEMP\_ADDR, 239
  - APP\_PB\_VBATT\_DEBOUNCE\_IN\_MS, 239
  - APP\_PR\_SWAP\_PENDING, 239
  - APP\_PSINK\_DIS\_MONITOR\_TIMER\_PERIOD, 239
  - APP\_PSINK\_DIS\_TIMER\_PERIOD, 239
  - APP\_PSINK\_DIS\_VBUS\_IN\_DIS\_PERIOD, 239
  - APP\_PSOURCE\_CF\_TIMER\_PERIOD, 240
  - APP\_PSOURCE\_DIS\_EXT\_DIS\_TIMER\_PERIOD, 240
  - APP\_PSOURCE\_SAFE\_FET\_ON\_MONITOR\_TIMER\_PERIOD, 240
  - APP\_PSOURCE\_VBUS\_SET\_TIMER\_PERIOD, 240
  - APP\_RESET\_VDM\_TIMER\_PERIOD, 240

- APP\_RETIMER\_DISABLE\_DELAY, 240
- APP\_RETIMER\_ENABLE\_DELAY, 240
- APP\_SBU\_DELAYED\_CONNECT\_PERIOD, 240
- APP\_UFP\_RECOV\_VCONN\_SWAP\_TIMER\_PERIOD, 240
- APP\_VCONN\_RECOVERY\_PERIOD, 241
- APP\_VCONN\_SWAP\_PENDING, 241
- APP\_VDM\_BUSY\_TIMER\_PERIOD, 241
- APP\_VDM\_FAIL\_RETRY\_PERIOD, 241
- app\_bc\_12\_sm\_start, 244
- app\_conf\_for\_faulty\_dev\_removal, 244
- app\_connect\_change\_handler, 245
- app\_contract\_handler, 245
- app\_disable\_pd\_port, 245
- app\_event\_handler, 246
- app\_get\_callback\_ptr, 246
- app\_get\_resp\_buf, 246
- app\_get\_status, 248
- app\_init, 248
- app\_is\_host\_hpd\_virtual, 248
- app\_is\_port\_enabled, 249
- app\_nb\_sys\_pwr\_state\_t, 243
- app\_otp\_check\_temp, 249
- app\_otp\_enable, 249
- app\_otp\_status, 249
- app\_ovp\_disable, 250
- app\_ovp\_enable, 250
- app\_port\_fault\_count\_exceeded, 251
- app\_port\_fault\_status\_mask\_t, 243
- app\_power\_share\_init, 251
- app\_sleep, 251
- app\_task, 251
- app\_thermistor\_type\_t, 243
- app\_update\_bc\_src\_support, 252
- app\_update\_sys\_pwr\_state, 252
- app\_uvp\_disable, 252
- app\_uvp\_enable, 253
- app\_validate\_configtable\_offsets, 253
- app\_vdm\_layer\_reset, 253
- app\_wakeup, 254
- CCG\_ACTIVITY\_TIMER\_PERIOD, 241
- cpg\_app\_task\_init, 254
- fault\_event\_handler, 254
- fault\_handler\_clear\_counts, 254
- fault\_handler\_init\_vars, 255
- fault\_handler\_task, 255
- ICL\_VSYS\_STABLE\_WAIT\_TIME, 241
- mux\_ctrl\_bb\_enable, 255
- mux\_ctrl\_init, 256
- mux\_ctrl\_set\_cfg, 256
- mux\_poll\_fnc\_cbk\_t, 242
- OTP\_DEBOUNCE\_PERIOD, 241
- PB\_DEBOUNCE\_PERIOD, 241
- pd\_get\_ptr\_cfg\_sub\_tbl, 256
- RIDGE\_INIT\_HPD\_DEQUEUE\_TIMER\_PERIOD, 241
- sln\_pd\_event\_handler, 257
- sys\_hw\_error\_type\_t, 244
- system\_sleep, 257
- system\_vconn\_ocp\_dis, 257
- system\_vconn\_ocp\_en, 258
- TBT\_MODE\_EXIT\_CHECK\_PERIOD, 242
- THROTTLE\_DEBOUNCE\_PERIOD, 242
- THROTTLE\_WAIT\_FOR\_PD\_PERIOD, 242
- TYPE\_A\_CUR\_SENSE\_TIMER\_PERIOD, 242
- TYPE\_A\_PWM\_STEP\_TIMER\_PERIOD, 242
- TYPE\_A\_REG\_SWITCH\_TIMER\_PERIOD, 242
- UCSI\_CONNECT\_EVENT\_PERIOD, 242
- vbus\_discharge\_off, 258
- vbus\_discharge\_on, 258
- vbus\_get\_value, 259
- vbus\_is\_present, 259
- vconn\_change\_handler, 259
- vconn\_disable, 261
- vconn\_enable, 261
- vconn\_is\_present, 261
- app/alt\_mode/alt\_mode\_hw.h, 195
- app/alt\_mode/alt\_modes\_mgr.h, 201
- app/alt\_mode/custom\_hpi\_vid.h, 216
- app/alt\_mode/dp\_sid.h, 218
- app/alt\_mode/vdm\_task\_mgr.h, 225
- app/app.h, 232
- app/battery\_charging.h, 263
- app/intel\_tbt/bb\_retimer.h, 275
- app/intel\_tbt/icl.h, 285
- app/intel\_tbt/intel\_ridge.h, 289
- app/intel\_tbt/intel\_vid.h, 293
- app/intel\_tbt/ridge\_slave.h, 296
- app/pdo.h, 302
- app/psink.h, 305
- app/psource.h, 307
- app/swap.h, 309
- app/vdm.h, 311
- app\_bc\_12\_sm\_start
  - app.h, 244
- app\_cbk
  - dpm\_status\_t, 75
- app\_cbk\_t, 38
  - app\_event\_handler, 39
  - eval\_dr\_swap, 39
  - eval\_enter\_usb, 39
  - eval\_fr\_swap, 39
  - eval\_pr\_swap, 39
  - eval\_rdo, 39
  - eval\_src\_cap, 39
  - eval\_vconn\_swap, 39
  - eval\_vdm, 40
  - psnk\_disable, 40
  - psnk\_enable, 40
  - psnk\_set\_current, 40
  - psnk\_set\_voltage, 40
  - psrc\_disable, 40
  - psrc\_enable, 40
  - psrc\_get\_voltage, 40
  - psrc\_set\_current, 40
  - psrc\_set\_voltage, 41

- vbus\_discharge\_off, 41
- vbus\_discharge\_on, 41
- vbus\_get\_value, 41
- vbus\_is\_present, 41
- vconn\_disable, 41
- vconn\_enable, 41
- vconn\_is\_present, 41
- app\_conf\_for\_faulty\_dev\_removal
  - app.h, 244
- app\_connect\_change\_handler
  - app.h, 245
- app\_contract\_handler
  - app.h, 245
- app\_disable\_pd\_port
  - app.h, 245
- app\_event\_handler
  - app.h, 246
  - app\_cbk\_t, 39
- app\_evt
  - alt\_mode\_reg\_info\_t, 37
- app\_evt\_data
  - alt\_mode\_info\_t, 35
- app\_evt\_needed
  - alt\_mode\_info\_t, 35
- app\_evt\_t
  - pd.h, 384
- app\_fault\_mask\_t
  - pd.h, 386
- app\_get\_callback\_ptr
  - app.h, 246
- app\_get\_resp\_buf
  - app.h, 246
- app\_get\_status
  - app.h, 248
- app\_init
  - app.h, 248
- app\_is\_host\_hpd\_virtual
  - app.h, 248
- app\_is\_port\_enabled
  - app.h, 249
- app\_nb\_sys\_pwr\_state\_t
  - app.h, 243
- app\_otp\_check\_temp
  - app.h, 249
- app\_otp\_enable
  - app.h, 249
- app\_otp\_status
  - app.h, 249
- app\_ovp\_disable
  - app.h, 250
- app\_ovp\_enable
  - app.h, 250
- app\_pending\_swaps
  - app\_status\_t, 44
- app\_port\_fault\_count\_exceeded
  - app.h, 251
- app\_port\_fault\_status\_mask\_t
  - app.h, 243
- app\_power\_share\_init
  - app.h, 251
- app\_req\_status\_t
  - pd.h, 386
- app\_resp
  - app\_status\_t, 44
- app\_resp\_cbk\_t
  - pd.h, 382
- app\_resp\_t, 42
  - req\_status, 42
  - resp\_do, 42
- app\_sleep
  - app.h, 251
- app\_status\_t, 42
  - actv\_swap\_count, 43
  - actv\_swap\_delay, 43
  - actv\_swap\_type, 44
  - alt\_mode\_entered, 44
  - alt\_mode\_trig\_mask, 44
  - app\_pending\_swaps, 44
  - app\_resp, 44
  - bc\_12\_src\_disabled, 44
  - cable\_retimer\_supp, 44
  - cbl\_disc\_id\_finished, 44
  - cbl\_rst\_done, 44
  - cur\_fb\_enabled, 45
  - custom\_hpi\_svid, 45
  - debug\_acc\_attached, 45
  - dfp\_alt\_mode\_mask, 45
  - disc\_cbl\_pending, 45
  - fault\_status, 45
  - is\_hot\_shutdown, 45
  - is\_mux\_busy, 45
  - is\_vbus\_on, 45
  - is\_vconn\_on, 46
  - is\_vdm\_pending, 46
  - keep\_vconn\_src, 46
  - ld\_sw\_ctrl, 46
  - mux\_poll\_cbk, 46
  - psnk\_cur, 46
  - psnk\_volt, 46
  - psrc\_rising, 46
  - psrc\_volt, 46
  - psrc\_volt\_old, 47
  - pwr\_ready\_cbk, 47
  - retimer\_dis\_req, 47
  - skip\_mux\_config, 47
  - snk\_dis\_cbk, 47
  - trig\_cbl\_rst, 47
  - turn\_off\_temp\_limit, 47
  - turn\_on\_temp\_limit, 47
  - ufp\_alt\_mode\_mask, 47
  - usb2\_supp, 48
  - usb3\_supp, 48
  - usb4\_active, 48
  - usb4\_data\_rst\_cnt, 48
  - vdm\_prcs\_failed, 48
  - vdm\_resp, 48



- vdm\_resp\_cbk, 48
- vdm\_retry\_pending, 48
- vdm\_task\_en, 48
- vdm\_version, 49
- app\_swap\_resp\_t
  - pd.h, 387
- app\_task
  - app.h, 251
- app\_thermistor\_type\_t
  - app.h, 243
- app\_update\_bc\_src\_support
  - app.h, 252
- app\_update\_rdo
  - pdo.h, 302
- app\_update\_sys\_pwr\_state
  - app.h, 252
- app\_uvp\_disable
  - app.h, 252
- app\_uvp\_enable
  - app.h, 253
- app\_validate\_configtable\_offsets
  - app.h, 253
- app\_vdm\_layer\_reset
  - app.h, 253
- app\_wakeup
  - app.h, 254
- apple\_src\_id
  - chg\_cfg\_params\_t, 63
- application
  - pd\_config\_t, 109
- apply\_threshold
  - utils.h, 571
- atch\_tgt\_info
  - alt\_mode\_reg\_info\_t, 37
- atch\_tgt\_info\_t, 49
  - ama\_vdo, 49
  - cbl\_svid, 49
  - cbl\_vdo, 49
  - tgt\_id\_header, 49
  - tgt\_svid, 50
- atch\_type
  - alt\_mode\_reg\_info\_t, 37
- attach
  - bc\_status\_t, 59
  - dpm\_status\_t, 76
- attached\_dev
  - dpm\_status\_t, 76
- auto\_cfg\_settings\_t, 50
  - policy\_mgr\_en, 50
  - port\_pwr, 50
  - pps\_en, 50
  - reserved\_0, 51
  - sensor\_data, 51
  - sys\_pwr, 51
  - table\_len, 51
  - unconstrained\_pwr\_en, 51
  - vin\_oc1, 51
  - vin\_oc2, 51
  - vin\_oc3, 51
  - vin\_throttling\_ctrl, 51
- auto\_cfg\_tbl\_offset
  - pd\_port\_config\_t, 125
- aux\_resistor\_config\_t
  - pdss\_hal.h, 449
- aux\_resistor\_configure
  - pdss\_hal.h, 455
- avoid\_retry
  - pd\_status\_t, 140
- b22\_retimer\_cbl
  - pd\_do\_t::TBT\_CBL\_VDO, 181
  - pd\_do\_t::TBT\_VDO, 183
- BB\_CONN\_STATE\_REG
  - bb\_retimer.h, 276
- BB\_DBR\_DEBUG\_MODE\_REG\_WRITE\_DELAY
  - bb\_retimer.h, 276
- BB\_DBR\_DEBUG\_POLL\_DELAY
  - bb\_retimer.h, 277
- BB\_DBR\_WAKEUP\_DELAY
  - bb\_retimer.h, 277
- BB\_DEBUG\_MODE\_CLEAR
  - bb\_retimer.h, 277
- BB\_DEBUG\_MODE\_DEFAULT
  - bb\_retimer.h, 277
- BB\_DEBUG\_MODE\_REG
  - bb\_retimer.h, 277
- BB\_DEBUG\_MODE\_RX\_LOCKED
  - bb\_retimer.h, 277
- BB\_DEBUG\_MODE\_SIZE
  - bb\_retimer.h, 277
- BB\_DEBUG\_MODE\_USB\_COMPLIANCE
  - bb\_retimer.h, 277
- BB\_DEBUGMODE\_POLL\_COUNT
  - bb\_retimer.h, 277
- BB\_IRQ\_HPD\_MASK
  - bb\_retimer.h, 278
- BB\_STATUS\_MASK
  - bb\_retimer.h, 278
- BB\_STATUS\_REG
  - bb\_retimer.h, 278
- BB\_STATUS\_Sx\_ACTIVE
  - bb\_retimer.h, 278
- BB\_STATUS
  - intel\_vid.h, 294
- BB\_USB\_3\_SPEED\_MASK
  - bb\_retimer.h, 278
- BC\_AMP\_LIMIT
  - battery\_charging.h, 265
- BC\_CMP\_0\_IDX
  - battery\_charging.h, 265
- BC\_CMP\_1\_IDX
  - battery\_charging.h, 265
- BC\_SINK\_1\_2\_MODE\_ENABLE\_MASK
  - pd.h, 363
- BC\_SINK\_APPLE\_MODE\_ENABLE\_MASK
  - pd.h, 363
- BC\_SRC\_1\_2\_MODE\_ENABLE\_MASK

- pd.h, 363
- BC\_SRC\_AFC\_MODE\_ENABLE\_MASK
  - pd.h, 363
- BC\_SRC\_APPLE\_MODE\_ENABLE\_MASK
  - pd.h, 363
- BC\_SRC\_QC\_4\_0\_MODE\_ENABLE\_MASK
  - pd.h, 363
- BC\_SRC\_QC\_MODE\_ENABLE\_MASK
  - pd.h, 363
- BC\_SRC\_QC\_VER\_2\_CLASS\_A\_VAL
  - pd.h, 364
- BC\_SRC\_QC\_VER\_2\_CLASS\_B\_VAL
  - pd.h, 364
- BC\_SRC\_QC\_VER\_3\_CLASS\_A\_VAL
  - pd.h, 364
- BC\_SRC\_QC\_VER\_3\_CLASS\_B\_VAL
  - pd.h, 364
- BDO\_HDR\_IDX
  - pd.h, 364
- BUSY\_WAIT\_US
  - utils.h, 567
- BYTE\_GET\_LOWER\_NIBBLE
  - utils.h, 567
- BYTE\_GET\_UPPER\_NIBBLE
  - utils.h, 567
- bat\_chg\_params\_t, 52
  - reserved\_0, 52
  - reserved\_1, 52
  - table\_len, 52
  - vbatt\_cutoff\_volt, 52
  - vbatt\_dischg\_en\_volt, 53
  - vbatt\_max\_cur, 53
  - vbatt\_max\_volt, 53
- bat\_chg\_tbl\_offset
  - pd\_port\_config\_t, 125
- bat\_snk
  - pd\_do\_t, 114
- bat\_src
  - pd\_do\_t, 114
- bat\_status\_change
  - pd\_do\_t::ADO\_ALERT, 28
- battery\_charging.h
  - APPLE\_AMP\_1A, 264
  - APPLE\_AMP\_2\_1A, 264
  - APPLE\_AMP\_2\_4A, 265
  - BC\_AMP\_LIMIT, 265
  - BC\_CMP\_0\_IDX, 265
  - BC\_CMP\_1\_IDX, 265
  - bc\_afc\_form\_vi, 270
  - bc\_apple\_brick\_id, 266
  - bc\_apple\_id\_t, 267
  - bc\_apple\_term, 267
  - bc\_charge\_mode\_t, 267
  - bc\_clear\_bc\_evt, 270
  - bc\_d\_status\_t, 268
  - bc\_fsm, 271
  - bc\_get\_config, 271
  - bc\_get\_status, 271
  - bc\_init, 272
  - bc\_is\_active, 272
  - bc\_pd\_event\_handler, 272
  - bc\_port\_is\_cdp, 273
  - bc\_port\_role\_t, 268
  - bc\_port\_type\_t, 268
  - bc\_qc\_class\_t, 269
  - bc\_qc\_ver\_t, 269
  - bc\_set\_bc\_evt, 273
  - bc\_sink\_timer\_t, 269
  - bc\_sleep, 273
  - bc\_start, 274
  - bc\_state\_t, 269
  - bc\_stop, 274
  - bc\_wakeup, 274
  - CCG\_POWER\_PRECISION\_MULT, 265
  - cpg\_get\_system\_max\_pdp, 274
  - QC3\_MIN\_VOLT, 265
  - QC\_AMP\_12V, 266
  - QC\_AMP\_20V, 266
  - QC\_AMP\_5V, 266
  - QC\_AMP\_9V, 266
  - QC\_AMP\_CONT, 266
  - QC\_CONT\_VOLT\_CHANGE\_PER\_PULSE, 266
  - qc\_set\_cf\_limit, 275
- battery\_input
  - pd\_power\_status\_t, 137
- bb\_always\_on
  - bb\_settings\_t, 55
- bb\_bos\_dscr\_offset
  - bb\_settings\_t, 55
- bb\_bus\_power
  - bb\_settings\_t, 56
- bb\_ec\_present
  - bb\_settings\_t, 56
- bb\_enable
  - bb\_settings\_t, 56
- bb\_option
  - bb\_settings\_t, 56
- bb\_pid
  - bb\_settings\_t, 56
- bb\_retimer.h
  - BB\_CONN\_STATE\_REG, 276
  - BB\_DBR\_DEBUG\_MODE\_REG\_WRITE\_DELAY, 276
  - BB\_DBR\_DEBUG\_POLL\_DELAY, 277
  - BB\_DBR\_WAKEUP\_DELAY, 277
  - BB\_DEBUG\_MODE\_CLEAR, 277
  - BB\_DEBUG\_MODE\_DEFAULT, 277
  - BB\_DEBUG\_MODE\_REG, 277
  - BB\_DEBUG\_MODE\_RX\_LOCKED, 277
  - BB\_DEBUG\_MODE\_SIZE, 277
  - BB\_DEBUG\_MODE\_USB\_COMPLIANCE, 277
  - BB\_DEBUGMODE\_POLL\_COUNT, 277
  - BB\_IRQ\_HPD\_MASK, 278
  - BB\_STATUS\_MASK, 278
  - BB\_STATUS\_REG, 278
  - BB\_STATUS\_Sx\_ACTIVE, 278

- BB\_USB\_3\_SPEED\_MASK, 278
- MAX\_NO\_OF\_RETIMERS, 278
- RETIMER\_CFG\_AVAILABLE, 278
- RETIMER\_CFG\_SLAVE\_ADDR, 278
- RETIMER\_I2C\_TIMEOUT, 278
- retimer\_clr\_evt, 279
- retimer\_disable, 279
- retimer\_enable, 280
- retimer\_force\_enable, 280
- retimer\_init, 280
- retimer\_is\_present, 281
- retimer\_read\_wrapper, 281
- retimer\_reg\_write, 281
- retimer\_set\_compl\_mode, 282
- retimer\_set\_evt, 282
- retimer\_set\_slave\_address, 282
- retimer\_sleep\_allowed, 283
- retimer\_start\_debug\_poll, 283
- retimer\_status\_update, 283
- retimer\_task, 284
- retimer\_update\_is\_pending, 284
- retimer\_write\_wrapper, 284
- rt\_evt\_t, 279
- set\_retimer\_status, 285
- bb\_settings\_t, 55
  - bb\_always\_on, 55
  - bb\_bos\_dscr\_offset, 55
  - bb\_bus\_power, 56
  - bb\_ec\_present, 56
  - bb\_enable, 56
  - bb\_option, 56
  - bb\_pid, 56
  - bb\_string\_dscr\_offset, 56
  - bb\_timeout, 56
  - bb\_unique\_container\_id, 57
  - bb\_vid, 57
  - reserved\_1, 57
  - table\_len, 57
- bb\_string\_dscr\_offset
  - bb\_settings\_t, 56
- bb\_tbl\_offset
  - pd\_port\_config\_t, 126
- bb\_timeout
  - bb\_settings\_t, 56
- bb\_unique\_container\_id
  - bb\_settings\_t, 57
- bb\_vid
  - bb\_settings\_t, 57
- bc\_12\_src\_disabled
  - app\_status\_t, 44
- bc\_afc\_form\_vi
  - battery\_charging.h, 270
- bc\_apple\_brick\_id
  - battery\_charging.h, 266
- bc\_apple\_id\_t
  - battery\_charging.h, 267
- bc\_apple\_term
  - battery\_charging.h, 267
- bc\_charge\_mode\_t
  - battery\_charging.h, 267
- bc\_clear\_bc\_evt
  - battery\_charging.h, 270
- bc\_d\_status\_t
  - battery\_charging.h, 268
- bc\_dp\_dm\_state\_t, 57
  - d, 57
  - state, 57
- bc\_evt
  - bc\_status\_t, 59
- bc\_fsm
  - battery\_charging.h, 271
- bc\_fsm\_state
  - bc\_status\_t, 59
- bc\_get\_config
  - battery\_charging.h, 271
- bc\_get\_status
  - battery\_charging.h, 271
- bc\_init
  - battery\_charging.h, 272
- bc\_is\_active
  - battery\_charging.h, 272
- bc\_pd\_event\_handler
  - battery\_charging.h, 272
- bc\_port\_is\_cdp
  - battery\_charging.h, 273
- bc\_port\_role\_t
  - battery\_charging.h, 268
- bc\_port\_type\_t
  - battery\_charging.h, 268
- bc\_qc\_class\_t
  - battery\_charging.h, 269
- bc\_qc\_ver\_t
  - battery\_charging.h, 269
- bc\_set\_bc\_evt
  - battery\_charging.h, 273
- bc\_sink\_timer\_t
  - battery\_charging.h, 269
- bc\_sleep
  - battery\_charging.h, 273
- bc\_start
  - battery\_charging.h, 274
- bc\_state\_t
  - battery\_charging.h, 269
- bc\_status\_t, 58
  - afc\_src\_cur\_match\_byte, 58
  - afc\_src\_is\_matched, 58
  - afc\_src\_last\_match\_byte, 58
  - afc\_src\_match\_count, 59
  - afc\_src\_matched\_byte, 59
  - afc\_src\_msg\_count, 59
  - afc\_tx\_active, 59
  - attach, 59
  - bc\_evt, 59
  - bc\_fsm\_state, 59
  - comp\_rising, 59
  - connected, 59

- cur\_amp, 60
- cur\_mode, 60
- cur\_timer, 60
- cur\_volt, 60
- dp\_dm\_status, 60
- old\_dp\_dm\_status, 60
- bc\_stop
  - battery\_charging.h, 274
- bc\_wakeup
  - battery\_charging.h, 274
- bcd\_dev
  - pd\_do\_t::STD\_PROD\_VDO, 174
- bist\_cm2\_enabled
  - dpm\_status\_t, 76
- bist\_do
  - pd\_do\_t, 114
- bist\_mode\_t
  - pd.h, 387
- bist\_test\_en
  - pd\_status\_t, 140
- boot.h
  - boot\_check\_for\_valid\_fw, 521
  - boot\_get\_boot\_seq, 521
  - boot\_get\_wait\_time, 522
  - boot\_handle\_validate\_fw\_cmd, 522
  - boot\_jump\_to\_fw, 522
  - boot\_start, 523
  - boot\_update\_fw\_status, 523
  - boot\_validate\_configtable, 523
  - boot\_validate\_fw, 524
  - CCG\_BL\_WAIT\_DEFAULT, 519
  - CCG\_BL\_WAIT\_MAXIMUM, 520
  - CCG\_BL\_WAIT\_MINIMUM, 520
  - CCG\_BOOT\_MODE\_RQT\_SIG, 520
  - CCG\_FW1\_BOOT\_RQT\_SIG, 520
  - CCG\_FW2\_BOOT\_RQT\_SIG, 520
  - CCG\_FW\_METADATA\_BOOTSEQ\_OFFSET, 520
  - CCG\_FWMETA\_APPID\_WAIT\_0, 520
  - CCG\_FWMETA\_APPID\_WAIT\_DEF, 520
  - CONFIGTABLE\_CHECKSUM\_OFFSET, 520
  - CONFIGTABLE\_CHECKSUM\_START, 521
  - CONFIGTABLE\_SIGNATURE, 521
  - CONFIGTABLE\_SIZE\_OFFSET, 521
  - CY\_PD\_IMG1\_FW\_STATUS\_BIT\_MASK, 521
  - get\_boot\_mode\_reason, 524
  - gl\_img1\_fw\_metadata, 524
  - gl\_img1\_fw\_pseudo\_metadata, 524
  - gl\_img2\_fw\_metadata, 525
  - gl\_img2\_fw\_pseudo\_metadata, 525
  - gl\_img\_status, 525
- boot\_app\_id
  - sys\_fw\_metadata\_t, 179
- boot\_app\_ver\_status
  - sys\_fw\_metadata\_t, 179
- boot\_app\_version
  - sys\_fw\_metadata\_t, 179
- boot\_check\_for\_valid\_fw
  - boot.h, 521
- boot\_get\_boot\_seq
  - boot.h, 521
- boot\_get\_wait\_time
  - boot.h, 522
- boot\_handle\_validate\_fw\_cmd
  - boot.h, 522
- boot\_jump\_to\_fw
  - boot.h, 522
- boot\_last\_row
  - sys\_fw\_metadata\_t, 179
- boot\_mode\_request
  - fw\_img\_status\_t::fw\_mode\_reason\_t, 97
- boot\_seq
  - sys\_fw\_metadata\_t, 179
- boot\_start
  - boot.h, 523
- boot\_update\_fw\_status
  - boot.h, 523
- boot\_validate\_configtable
  - boot.h, 523
- boot\_validate\_fw
  - boot.h, 524
- bootup
  - dpm\_status\_t, 76
- buf\_size
  - i2c\_scb\_config\_t, 98
- buffer
  - i2c\_scb\_config\_t, 98
- CABLE\_CURR\_3A
  - ucsi.h, 577
- CABLE\_CURR\_5A
  - ucsi.h, 578
- CABLE\_CURR\_DFLT
  - ucsi.h, 578
- CABLE\_SPEED\_10GBPS
  - ucsi.h, 578
- CABLE\_SPEED\_480MBPS
  - ucsi.h, 578
- CABLE\_SPEED\_5GBPS
  - ucsi.h, 578
- CABLE\_VDO\_DIRECTION
  - ucsi.h, 578
- CABLE
  - alt\_modes\_mgr.h, 204
- CALL\_MAP
  - srom.h, 545
- CBL\_DIR\_SUPP\_MASK
  - alt\_modes\_mgr.h, 204
- CC\_CHANNEL\_1
  - pd.h, 364
- CC\_CHANNEL\_2
  - pd.h, 364
- CCG\_ACTIVITY\_TIMER\_PERIOD
  - app.h, 241
- CCG\_BL\_WAIT\_DEFAULT
  - boot.h, 519
- CCG\_BL\_WAIT\_MAXIMUM
  - boot.h, 520

- CCG\_BL\_WAIT\_MINIMUM  
boot.h, 520
- CCG\_BOOT\_MODE\_RQT\_SIG  
boot.h, 520
- CCG\_CC\_STAT\_DRP\_TOGGLE  
pd.h, 364
- CCG\_CC\_STAT\_RD\_PRESENT  
pd.h, 364
- CCG\_CC\_STAT\_RP\_PRESENT  
pd.h, 365
- CCG\_CC\_STAT\_VCONN\_ACTIVE  
pd.h, 365
- CCG\_CC\_STAT\_ZOPEN  
pd.h, 365
- CCG\_DPM\_ERROR\_NO\_VCONN  
pd.h, 365
- CCG\_DPM\_ERROR\_NONE  
pd.h, 365
- CCG\_FRS\_RX\_ENABLE\_MASK  
pd.h, 365
- CCG\_FRS\_TX\_ENABLE\_MASK  
pd.h, 365
- CCG\_FW1\_BOOT\_RQT\_SIG  
boot.h, 520
- CCG\_FW2\_BOOT\_RQT\_SIG  
boot.h, 520
- CCG\_FW\_METADATA\_BOOTSEQ\_OFFSET  
boot.h, 520
- CCG\_FWMETA\_APPID\_WAIT\_0  
boot.h, 520
- CCG\_FWMETA\_APPID\_WAIT\_DEF  
boot.h, 520
- CCG\_PD\_EXT\_PPS\_STATUS\_SIZE  
pd.h, 365
- CCG\_PD\_EXT\_SNKCAP\_BUF\_SIZE  
pd.h, 365
- CCG\_PD\_EXT\_SNKCAP\_SIZE  
pd.h, 366
- CCG\_PD\_EXT\_SNKCAP\_VERS\_INDEX  
pd.h, 366
- CCG\_PD\_EXT\_SRCCAP\_INP\_INDEX  
pd.h, 366
- CCG\_PD\_EXT\_SRCCAP\_INP\_UNCONSTRAINED  
pd.h, 366
- CCG\_PD\_EXT\_SRCCAP\_PDP\_INDEX  
pd.h, 366
- CCG\_PD\_EXT\_SRCCAP\_SIZE  
pd.h, 366
- CCG\_PD\_EXT\_STATUS\_SIZE  
pd.h, 366
- CCG\_PD\_FIX\_SRC\_PDO\_MASK\_REV2  
pd.h, 366
- CCG\_PD\_FIX\_SRC\_PDO\_MASK\_REV3  
pd.h, 366
- CCG\_PD\_FLAG\_CONTRACT\_NEG\_ACTIVE  
pd.h, 367
- CCG\_PD\_FLAG\_EXPLICIT\_CONTRACT  
pd.h, 367
- CCG\_PD\_FLAG\_POWER\_SINK  
pd.h, 367
- CCG\_PD\_FLAG\_SRC\_READY  
pd.h, 367
- CCG\_POWER\_PRECISION\_MULT  
battery\_charging.h, 265
- CCG\_STATUS\_CODE\_OFFSET  
status.h, 546
- CCG\_STATUS\_TO\_HPI\_RESPONSE  
status.h, 546
- CCG\_USB4\_EUDO\_USB4\_EN\_MASK  
pd.h, 367
- CCG\_USB\_MODE\_USB2  
pd.h, 367
- CCG\_USB\_MODE\_USB3  
pd.h, 367
- CCG\_USB\_MODE\_USB4  
pd.h, 367
- CERT\_STAT\_IDX  
pd.h, 367
- CONFIGTABLE\_CHECKSUM\_OFFSET  
boot.h, 520
- CONFIGTABLE\_CHECKSUM\_START  
boot.h, 521
- CONFIGTABLE\_SIGNATURE  
boot.h, 521
- CONFIGTABLE\_SIZE\_OFFSET  
boot.h, 521
- CY\_PD\_IMG1\_FW\_STATUS\_BIT\_MASK  
boot.h, 521
- CY\_VID  
pd.h, 368
- cable\_active  
pd\_do\_t::TBT\_VDO, 183
- cable\_current  
pd\_do\_t::ENTERUSB\_VDO, 90
- cable\_disc\_cnt  
pd\_config\_t, 109
- cable\_disc\_en  
pd\_port\_config\_t, 126
- cable\_resistance  
pwr\_params\_t, 147
- cable\_retimer\_supp  
app\_status\_t, 44
- cable\_speed  
pd\_do\_t::ENTERUSB\_VDO, 90
- cable\_type  
pd\_do\_t::ENTERUSB\_VDO, 90
- cap\_mismatch  
pd\_do\_t::RDO\_BAT\_GIVEBACK, 154  
pd\_do\_t::RDO\_BAT, 152  
pd\_do\_t::RDO\_FIXED\_VAR\_GIVEBACK, 157  
pd\_do\_t::RDO\_FIXED\_VAR, 156  
pd\_do\_t::RDO\_GEN\_GVB, 161  
pd\_do\_t::RDO\_GEN, 159  
pd\_do\_t::RDO\_PPS, 162
- cb  
ccg\_timer\_t, 62

cb\_fun\_ptr  
     i2c\_scb\_config\_t, 98  
 cbk  
     alt\_mode\_info\_t, 35  
     pd\_status\_t, 140  
 cbl\_disc\_id\_finished  
     app\_status\_t, 44  
 cbl\_dsc  
     dpm\_status\_t, 76  
 cbl\_fw\_ver  
     pd\_do\_t::ACT\_CBL\_VDO\_1, 24  
     pd\_do\_t::ACT\_CBL\_VDO, 21  
     pd\_do\_t::PAS\_CBL\_VDO, 105  
     pd\_do\_t::STD\_CBL\_VDO, 170  
 cbl\_gen  
     pd\_do\_t::TBT\_CBL\_VDO, 181  
     pd\_do\_t::TBT\_VDO, 183  
 cbl\_hw\_ver  
     pd\_do\_t::ACT\_CBL\_VDO\_1, 24  
     pd\_do\_t::ACT\_CBL\_VDO, 21  
     pd\_do\_t::PAS\_CBL\_VDO, 105  
     pd\_do\_t::STD\_CBL\_VDO, 171  
 cbl\_latency  
     pd\_do\_t::ACT\_CBL\_VDO\_1, 24  
     pd\_do\_t::ACT\_CBL\_VDO, 22  
     pd\_do\_t::PAS\_CBL\_VDO, 105  
     pd\_do\_t::STD\_CBL\_VDO, 171  
 cbl\_mode\_en  
     dpm\_status\_t, 76  
 cbl\_obj\_pos  
     alt\_mode\_info\_t, 35  
 cbl\_rst\_done  
     app\_status\_t, 44  
 cbl\_soft\_reset\_tried  
     dpm\_status\_t, 76  
 cbl\_sop\_flag  
     alt\_mode\_reg\_info\_t, 37  
 cbl\_speed  
     pd\_do\_t::TBT\_CBL\_VDO, 181  
     pd\_do\_t::TBT\_VDO, 183  
 cbl\_state  
     dpm\_status\_t, 76  
 cbl\_svid  
     atch\_tgt\_info\_t, 49  
 cbl\_term  
     pd\_do\_t::ACT\_CBL\_VDO\_1, 24  
     pd\_do\_t::ACT\_CBL\_VDO, 22  
     pd\_do\_t::PAS\_CBL\_VDO, 105  
     pd\_do\_t::STD\_CBL\_VDO, 171  
 cbl\_term\_t  
     pd.h, 387  
 cbl\_type  
     dpm\_status\_t, 76  
     pd\_do\_t::TBT\_CBL\_VDO, 181  
     pd\_do\_t::TBT\_VDO, 183  
 cbl\_type\_t  
     pd.h, 387  
 cbl\_vbus\_cur\_t  
     pd.h, 388  
 cbl\_vdm\_version  
     dpm\_status\_t, 77  
 cbl\_vdo  
     atch\_tgt\_info\_t, 49  
     dpm\_status\_t, 77  
 cbl\_vdo\_2  
     dpm\_status\_t, 77  
 cbl\_wait  
     dpm\_status\_t, 77  
 cc  
     cc\_state\_t, 61  
 cc\_live  
     dpm\_status\_t, 77  
 cc\_old\_status  
     dpm\_status\_t, 77  
 cc\_rd\_status  
     dpm\_status\_t, 77  
 cc\_state\_t, 61  
     cc, 61  
     state, 61  
 cc\_status  
     dpm\_status\_t, 77  
 ccg\_app\_task\_init  
     app.h, 254  
 ccg\_bc\_cdp\_en  
     pdss\_hal.h, 456  
 ccg\_bc\_cdp\_sm  
     pdss\_hal.h, 456  
 ccg\_bc\_dcp\_en  
     pdss\_hal.h, 456  
 ccg\_bc\_dis  
     pdss\_hal.h, 457  
 ccg\_bc\_is\_cdp  
     pdss\_hal.h, 457  
 ccg\_config\_dp\_dm\_mux  
     pdss\_hal.h, 457  
 ccg\_get\_system\_max\_pdp  
     battery\_charging.h, 274  
 ccg\_is\_cdp\_sm\_busy  
     pdss\_hal.h, 458  
 ccg\_refgen\_op\_t  
     pdss\_hal.h, 449  
 ccg\_set\_fault\_cb  
     pdss\_hal.h, 458  
 ccg\_spec\_rev  
     pd\_port\_status\_t::PD\_PORT\_STAT, 134  
 ccg\_status\_t  
     status.h, 546  
 ccg\_supply\_t  
     pdss\_hal.h, 450  
 ccg\_timer\_id\_t  
     timer\_id.h, 561  
 ccg\_timer\_t, 62  
     cb, 62  
     count, 62  
     id, 62  
     period, 62

- ccgx\_version.h
  - FW\_BASE\_VERSION, 525
- chg\_cfg\_params\_t, 63
  - afc\_src\_cap\_cnt, 63
  - afc\_src\_caps, 63
  - apple\_src\_id, 63
  - qc\_src\_type, 63
  - reserved\_0, 64
  - reserved\_1, 64
  - snk\_sel, 64
  - src\_sel, 64
  - table\_len, 64
- chg\_cfg\_tbl\_offset
  - pd\_port\_config\_t, 126
- chunk\_no
  - pd\_extd\_hdr\_t::EXTD\_HDR\_T, 92
  - pd\_hdr\_t::PD\_HDR, 119
- chunked
  - pd\_extd\_hdr\_t::EXTD\_HDR\_T, 92
  - pd\_hdr\_t::PD\_HDR, 119
- clock\_freq
  - i2c\_scb\_config\_t, 98
- cmd
  - pd\_do\_t::STD\_VDM\_HDR, 176
  - pd\_do\_t::USTD\_VDM\_HDR, 188
- cmd\_0
  - pd\_do\_t::USTD\_QC\_4\_0\_HDR, 188
- cmd\_1
  - pd\_do\_t::USTD\_QC\_4\_0\_HDR, 188
- cmd\_do
  - dpm\_pd\_cmd\_buf\_t, 72
- cmd\_p
  - dpm\_status\_t, 77
- cmd\_sop
  - dpm\_pd\_cmd\_buf\_t, 72
- cmd\_type
  - pd\_do\_t::STD\_VDM\_HDR, 176
  - pd\_do\_t::USTD\_VDM\_HDR, 189
- comp\_id\_t
  - pdss\_hal.h, 450
- comp\_rising
  - bc\_status\_t, 59
- comp\_tr\_id\_t
  - pdss\_hal.h, 450
- connect
  - dpm\_status\_t, 78
- connected
  - bc\_status\_t, 59
- contract
  - dpm\_status\_t, 78
- contract\_exist
  - dpm\_status\_t, 78
  - pd\_port\_status\_t::PD\_PORT\_STAT, 134
- contract\_t, 64
  - cur\_pwr, 65
  - max\_volt, 65
  - min\_volt, 65
- count
  - ccg\_timer\_t, 62
- crc16
  - utils.h, 571
- cs\_res
  - ocp\_settings\_t, 101
- ctrl\_msg\_t
  - pd.h, 388
- ctrs
  - pd\_status\_t, 140
- cur\_amp
  - bc\_status\_t, 60
- cur\_data\_role
  - pd\_port\_status\_t::PD\_PORT\_STAT, 134
- cur\_fb
  - dpm\_status\_t, 78
  - pd\_do\_t::PPS\_SNK, 143
- cur\_fb\_enabled
  - app\_status\_t, 45
- cur\_mode
  - bc\_status\_t, 60
- cur\_port\_role
  - dpm\_status\_t, 78
- cur\_port\_type
  - dpm\_status\_t, 78
- cur\_power\_role
  - pd\_port\_status\_t::PD\_PORT\_STAT, 134
- cur\_pwr
  - contract\_t, 65
- cur\_rec\_msg\_id
  - pd\_status\_t, 141
- cur\_sense\_res
  - pwr\_params\_t, 147
- cur\_snk\_max\_min
  - dpm\_status\_t, 78
- cur\_snk\_pdo
  - dpm\_status\_t, 78
- cur\_snk\_pdo\_count
  - dpm\_status\_t, 78
- cur\_src\_pdo
  - dpm\_status\_t, 79
- cur\_src\_pdo\_count
  - dpm\_status\_t, 79
- cur\_timer
  - bc\_status\_t, 60
- cur\_volt
  - bc\_status\_t, 60
- current\_level
  - pd\_port\_config\_t, 126
- custom\_alt\_cfg\_settings\_t, 65
  - custom\_alt\_mode, 65
  - custom\_dfp\_mask, 65
  - custom\_ufp\_mask, 66
  - reserved0, 66
  - reserved1, 66
  - table\_len, 66
- custom\_alt\_mode
  - custom\_alt\_cfg\_settings\_t, 65
- custom\_alt\_mode\_tbl\_offset

- pd\_port\_config\_t, 126
- custom\_att\_obj\_pos
  - alt\_mode\_info\_t, 35
- custom\_dfp\_mask
  - custom\_alt\_cfg\_settings\_t, 65
- custom\_host\_cfg\_settings\_t, 66
  - acc\_mode\_disable, 67
  - ext\_powered\_prs, 67
  - pdo\_sel\_alg, 67
  - pwr\_threshold, 67
  - req\_max\_pwr, 67
  - reserved, 67
  - rp\_detach\_disable, 67
  - snk\_path\_enable, 67
  - table\_len, 67
- custom\_host\_tbl\_offset
  - pd\_port\_config\_t, 126
- custom\_hpi\_svid
  - app\_status\_t, 45
- custom\_hpi\_vid.h
  - HPI\_AM\_SVID, 217
  - hpi\_am\_state\_t, 217
  - MAX\_HPI\_AM\_VDO\_NUMB, 217
  - reg\_hpi\_modes, 217
- custom\_ufp\_mask
  - custom\_alt\_cfg\_settings\_t, 66
- cutoff\_val
  - otp\_settings\_t, 102
- cutoff\_val\_1
  - otp\_settings\_t, 102
- d
  - bc\_dp\_dm\_state\_t, 57
- DATA\_RST\_RETRY\_NUMB
  - vdm\_task\_mngr.h, 227
- DFP\_ALT\_MODE\_HPI\_OFFSET
  - alt\_modes\_mngr.h, 204
- DFP\_D\_CONN
  - dp\_sid.h, 219
- DIV\_ROUND\_NEAREST
  - utils.h, 567
- DIV\_ROUND\_UP
  - utils.h, 567
- DP\_1\_3\_SIGNALING
  - dp\_sid.h, 219
- DP\_ALLOWED\_MUX\_CONFIG\_EVT
  - dp\_sid.h, 219
- DP\_ALT\_MODE\_ID
  - dp\_sid.h, 220
- DP\_APP\_CFG\_CMD\_MAX\_NUMB
  - dp\_sid.h, 220
- DP\_APP\_CFG\_CMD
  - dp\_sid.h, 220
- DP\_APP\_CFG\_USB\_IDX
  - dp\_sid.h, 220
- DP\_APP\_VCONN\_SWAP\_CFG\_CMD
  - dp\_sid.h, 220
- DP\_CFG\_CMD\_ACK\_MASK
  - dp\_sid.h, 220
- DP\_CONFIG\_SELECT\_DPSRC
  - dp\_sid.h, 220
- DP\_CONFIG\_SELECT
  - dp\_sid.h, 220
- DP\_DFP\_D\_CONFIG\_C
  - dp\_sid.h, 220
- DP\_DFP\_D\_CONFIG\_D
  - dp\_sid.h, 221
- DP\_DFP\_D\_CONFIG\_E
  - dp\_sid.h, 221
- DP\_DFP\_D\_CONFIG\_F
  - dp\_sid.h, 221
- DP\_HPD\_STATE\_MASK
  - dp\_sid.h, 221
- DP\_HPD\_TYPE\_GPIO
  - pd.h, 368
- DP\_HPD\_TYPE\_VIRTUAL
  - pd.h, 368
- DP\_INVALID\_CFG
  - dp\_sid.h, 221
- DP\_MAX\_IRQ\_SIZE
  - dp\_sid.h, 221
- DP\_MUX\_CTRL\_CMD
  - dp\_sid.h, 221
- DP\_QUEUE\_EMPTY\_INDEX
  - dp\_sid.h, 221
- DP\_QUEUE\_FULL\_INDEX
  - dp\_sid.h, 221
- DP\_QUEUE\_STATE\_SIZE
  - dp\_sid.h, 222
- DP\_SINK\_CTRL\_CMD
  - dp\_sid.h, 222
- DP\_STATUS\_UPDATE\_EVT
  - dp\_sid.h, 222
- DP\_SVID
  - dp\_sid.h, 222
  - pd.h, 368
- DP\_UFP\_MAX\_QUEUE\_SIZE
  - dp\_sid.h, 222
- DP\_USB\_SS\_CONFIG
  - dp\_sid.h, 222
- DP\_VDO\_IDX
  - dp\_sid.h, 222
- DRP\_TOGGLE\_PERIOD
  - pd.h, 368
- DWORD\_GET\_BYTE0
  - utils.h, 568
- DWORD\_GET\_BYTE1
  - utils.h, 568
- DWORD\_GET\_BYTE2
  - utils.h, 568
- DWORD\_GET\_BYTE3
  - utils.h, 568
- dat
  - pd\_packet\_extd\_t, 122
  - pd\_packet\_t, 123
- dat\_ptr
  - dpm\_pd\_cmd\_buf\_t, 72



- data\_0
  - pd\_do\_t::QC\_4\_0\_DATA\_VDO, 151
- data\_1
  - pd\_do\_t::QC\_4\_0\_DATA\_VDO, 151
- data\_2
  - pd\_do\_t::QC\_4\_0\_DATA\_VDO, 151
- data\_3
  - pd\_do\_t::QC\_4\_0\_DATA\_VDO, 151
- data\_msg\_t
  - pd.h, 389
- data\_reset\_state\_t
  - pd.h, 389
- data\_role
  - alt\_mode\_evt\_t::ALT\_MODE\_EVT, 31
  - alt\_mode\_hw\_evt\_t::ALT\_MODE\_HW\_EVT, 33
  - alt\_mode\_reg\_info\_t, 37
  - pd\_hdr\_t::PD\_HDR, 119
  - pd\_packet\_extd\_t, 122
  - pd\_packet\_t, 123
- data\_size
  - pd\_extd\_hdr\_t::EXTD\_HDR\_T, 92
  - pd\_hdr\_t::PD\_HDR, 119
- db\_event\_mask
  - pd\_config\_t, 109
- db\_support
  - dpm\_status\_t, 79
- dead\_bat
  - dpm\_status\_t, 79
- dead\_bat\_support
  - pd\_port\_config\_t, 126
- debounce
  - ocp\_settings\_t, 101
  - otp\_settings\_t, 103
  - ovp\_settings\_t, 104
  - scp\_settings\_t, 164
  - uvp\_settings\_t, 190
  - vconn\_ocp\_settings\_t, 192
- debounce2
  - ocp\_settings\_t, 101
- debounce\_ms
  - ovp\_settings\_t, 104
- debug\_acc\_attached
  - app\_status\_t, 45
- default\_role
  - pd\_port\_config\_t, 126
- default\_sink\_pdo\_mask
  - pd\_port\_config\_t, 126
- default\_src\_pdo\_mask
  - pd\_port\_config\_t, 127
- dev\_cap
  - pd\_do\_t::UFP\_VDO\_1, 187
- dflt\_data\_pref
  - pd\_port\_status\_t::PD\_PORT\_STAT, 134
- dflt\_data\_role
  - pd\_port\_status\_t::PD\_PORT\_STAT, 134
- dflt\_port\_role
  - dpm\_status\_t, 79
- dflt\_power\_pref
  - pd\_port\_status\_t::PD\_PORT\_STAT, 135
- dflt\_power\_role
  - pd\_port\_status\_t::PD\_PORT\_STAT, 135
- dfp\_alt\_mode\_mask
  - app\_status\_t, 45
- dfp\_asgmt
  - pd\_do\_t::DP\_CONFIG\_VDO, 69
- dfp\_d\_pin
  - pd\_do\_t::STD\_DP\_VDO, 173
- dfp\_mask
  - alt\_cfg\_settings\_t, 30
- dfp\_ufp\_conn
  - pd\_do\_t::DP\_STATUS\_VDO, 70
- dfp\_vdo
  - pd\_do\_t, 114
- disc\_cbl\_pending
  - app\_status\_t, 45
- div\_round\_up
  - utils.h, 571
- do\_count
  - vdm\_resp\_t, 194
- dock\_cfg\_tbl\_offset
  - pd\_port\_config\_t, 127
- dp\_cfg\_vdo
  - pd\_do\_t, 114
- dp\_config\_supported
  - pd\_port\_config\_t, 127
- dp\_conn\_t
  - dp\_sid.h, 224
- dp\_dm\_status
  - bc\_status\_t, 60
- dp\_mux\_control
  - pd\_port\_config\_t, 127
- dp\_oper
  - pd\_port\_config\_t, 127
- dp\_port\_cap\_t
  - dp\_sid.h, 224
- dp\_pref\_mode
  - pd\_port\_config\_t, 127
- dp\_sid.h
  - DFP\_D\_CONN, 219
  - DP\_1\_3\_SIGNALING, 219
  - DP\_ALLOWED\_MUX\_CONFIG\_EVT, 219
  - DP\_ALT\_MODE\_ID, 220
  - DP\_APP\_CFG\_CMD\_MAX\_NUMB, 220
  - DP\_APP\_CFG\_CMD, 220
  - DP\_APP\_CFG\_USB\_IDX, 220
  - DP\_APP\_VCONN\_SWAP\_CFG\_CMD, 220
  - DP\_CFG\_CMD\_ACK\_MASK, 220
  - DP\_CONFIG\_SELECT\_DPSRC, 220
  - DP\_CONFIG\_SELECT, 220
  - DP\_DFP\_D\_CONFIG\_C, 220
  - DP\_DFP\_D\_CONFIG\_D, 221
  - DP\_DFP\_D\_CONFIG\_E, 221
  - DP\_DFP\_D\_CONFIG\_F, 221
  - DP\_HPD\_STATE\_MASK, 221
  - DP\_INVALID\_CFG, 221
  - DP\_MAX\_IRQ\_SIZE, 221

- DP\_MUX\_CTRL\_CMD, 221
- DP\_QUEUE\_EMPTY\_INDEX, 221
- DP\_QUEUE\_FULL\_INDEX, 221
- DP\_QUEUE\_STATE\_SIZE, 222
- DP\_SINK\_CTRL\_CMD, 222
- DP\_STATUS\_UPDATE\_EVT, 222
- DP\_SVID, 222
- DP\_UFP\_MAX\_QUEUE\_SIZE, 222
- DP\_USB\_SS\_CONFIG, 222
- DP\_VDO\_IDX, 222
- dp\_conn\_t, 224
- dp\_port\_cap\_t, 224
- dp\_stat\_bm\_t, 224
- dp\_state\_t, 225
- GET\_HPD\_IRQ\_STAT, 222
- HPD\_HIGH\_IRQ\_HIGH, 222
- HPD\_HIGH\_IRQ\_LOW, 223
- HPD\_IRQ\_BIT\_POS, 223
- HPD\_LOW\_IRQ\_HIGH, 223
- HPD\_LOW\_IRQ\_LOW, 223
- HPD\_STATE\_BIT\_POS, 223
- MAX\_DP\_VDO\_NUMB, 223
- reg\_dp\_modes, 225
- STATUS\_UPDATE\_VDO, 223
- UFP\_D\_CONN, 223
- USB\_CONFIG\_SELECT, 223
- dp\_snk\_get\_hpd\_state
  - alt\_mode\_hw.h, 199
- dp\_stat\_bm\_t
  - dp\_sid.h, 224
- dp\_stat\_vdo
  - pd\_do\_t, 114
- dp\_state\_t
  - dp\_sid.h, 225
- dpdm\_mux\_cfg\_t
  - pdss\_hal.h, 451
- dpm.h
  - dpm\_clear\_fault\_active, 330
  - dpm\_clear\_hard\_reset\_count, 330
  - dpm\_deepsleep, 331
  - dpm\_disable, 331
  - dpm\_downgrade\_pd\_port\_rev, 331
  - dpm\_get\_cable\_usb\_cap, 332
  - dpm\_get\_def\_cable\_cap, 332
  - dpm\_get\_info, 332
  - dpm\_get\_mux\_enable\_wait\_period, 333
  - dpm\_get\_ndiscover\_identity\_count, 333
  - dpm\_get\_pd\_port\_status, 333
  - dpm\_get\_polarity, 333
  - dpm\_get\_rp\_audio\_accessory, 334
  - dpm\_get\_sink\_detach\_margin, 334
  - dpm\_get\_sink\_detach\_voltage, 334
  - dpm\_get\_snk\_wait\_cap\_period, 335
  - dpm\_get\_stack\_config, 335
  - dpm\_get\_vbus\_voltage, 335
  - dpm\_init, 335
  - dpm\_is\_accessory\_mode\_disabled, 336
  - dpm\_is\_idle, 336
  - dpm\_is\_rdo\_valid, 336
  - dpm\_is\_rp\_detach\_detect\_disabled, 337
  - dpm\_pd3\_src\_rp\_flow\_control, 337
  - dpm\_pd\_command, 337
  - dpm\_pd\_command\_ec, 339
  - dpm\_pe\_stop, 339
  - dpm\_pps\_task, 340
  - dpm\_prot\_reset, 340
  - dpm\_prot\_reset\_rx, 340
  - dpm\_refresh\_snk\_cap, 341
  - dpm\_refresh\_src\_cap, 341
  - dpm\_send\_hard\_reset, 341
  - dpm\_set\_accessory\_mode\_disabled, 342
  - dpm\_set\_alert, 342
  - dpm\_set\_cf, 342
  - dpm\_set\_chunk\_transfer\_running, 343
  - dpm\_set\_delay\_src\_cap\_start, 343
  - dpm\_set\_fault\_active, 344
  - dpm\_set\_rp\_detach\_detect\_disabled, 344
  - dpm\_sleep, 344
  - dpm\_start, 344
  - dpm\_stop, 346
  - dpm\_task, 346
  - dpm\_typec\_command, 346
  - dpm\_typec\_deassert\_rp\_rd, 347
  - dpm\_update\_def\_cable\_cap, 347
  - dpm\_update\_ext\_snk\_cap, 347
  - dpm\_update\_ext\_src\_cap, 348
  - dpm\_update\_frs\_enable, 348
  - dpm\_update\_mux\_enable\_wait\_period, 349
  - dpm\_update\_ndiscover\_identity\_count, 349
  - dpm\_update\_port\_config, 349
  - dpm\_update\_port\_status, 350
  - dpm\_update\_rp\_audio\_accessory, 350
  - dpm\_update\_snk\_cap, 351
  - dpm\_update\_snk\_cap\_mask, 351
  - dpm\_update\_snk\_max\_min, 351
  - dpm\_update\_snk\_wait\_cap\_period, 352
  - dpm\_update\_src\_cap, 352
  - dpm\_update\_src\_cap\_mask, 352
  - dpm\_update\_swap\_response, 353
  - dpm\_wakeup, 353
  - gl\_dpm\_port\_type, 353
- dpm\_clear\_fault\_active
  - dpm.h, 330
- dpm\_clear\_hard\_reset\_count
  - dpm.h, 330
- dpm\_cmd\_buf
  - dpm\_status\_t, 79
- dpm\_deepsleep
  - dpm.h, 331
- dpm\_disable
  - dpm.h, 331
- dpm\_downgrade\_pd\_port\_rev
  - dpm.h, 331
- dpm\_enabled
  - dpm\_status\_t, 79
- dpm\_err\_info

- dpm\_status\_t, 79
- dpm\_get\_cable\_usb\_cap
  - dpm.h, 332
- dpm\_get\_def\_cable\_cap
  - dpm.h, 332
- dpm\_get\_info
  - dpm.h, 332
- dpm\_get\_mux\_enable\_wait\_period
  - dpm.h, 333
- dpm\_get\_ndiscover\_identity\_count
  - dpm.h, 333
- dpm\_get\_pd\_port\_status
  - dpm.h, 333
- dpm\_get\_polarity
  - dpm.h, 333
- dpm\_get\_rp\_audio\_accessory
  - dpm.h, 334
- dpm\_get\_sink\_detach\_margin
  - dpm.h, 334
- dpm\_get\_sink\_detach\_voltage
  - dpm.h, 334
- dpm\_get\_snk\_wait\_cap\_period
  - dpm.h, 335
- dpm\_get\_stack\_config
  - dpm.h, 335
- dpm\_get\_vbus\_voltage
  - dpm.h, 335
- dpm\_init
  - dpm.h, 335
  - dpm\_status\_t, 79
- dpm\_is\_accessory\_mode\_disabled
  - dpm.h, 336
- dpm\_is\_idle
  - dpm.h, 336
- dpm\_is\_rdo\_valid
  - dpm.h, 336
- dpm\_is\_rp\_detach\_detect\_disabled
  - dpm.h, 337
- dpm\_pd3\_src\_rp\_flow\_control
  - dpm.h, 337
- dpm\_pd\_cbk
  - dpm\_status\_t, 80
- dpm\_pd\_cmd
  - dpm\_status\_t, 80
- dpm\_pd\_cmd\_active
  - dpm\_status\_t, 80
- dpm\_pd\_cmd\_buf\_t, 71
  - cmd\_do, 72
  - cmd\_sop, 72
  - dat\_ptr, 72
  - extd\_hdr, 72
  - extd\_type, 72
  - no\_of\_cmd\_do, 72
  - timeout, 72
- dpm\_pd\_cmd\_cbk\_t
  - pd.h, 382
- dpm\_pd\_cmd\_t
  - pd.h, 390
- dpm\_pd\_command
  - dpm.h, 337
- dpm\_pd\_command\_ec
  - dpm.h, 339
- dpm\_pe\_stop
  - dpm.h, 339
- dpm\_pps\_task
  - dpm.h, 340
- dpm\_prot\_reset
  - dpm.h, 340
- dpm\_prot\_reset\_rx
  - dpm.h, 340
- dpm\_refresh\_snk\_cap
  - dpm.h, 341
- dpm\_refresh\_src\_cap
  - dpm.h, 341
- dpm\_safe\_disable
  - dpm\_status\_t, 80
- dpm\_send\_hard\_reset
  - dpm.h, 341
- dpm\_set\_accessory\_mode\_disabled
  - dpm.h, 342
- dpm\_set\_alert
  - dpm.h, 342
- dpm\_set\_cf
  - dpm.h, 342
- dpm\_set\_chunk\_transfer\_running
  - dpm.h, 343
- dpm\_set\_delay\_src\_cap\_start
  - dpm.h, 343
- dpm\_set\_fault\_active
  - dpm.h, 344
- dpm\_set\_rp\_detach\_detect\_disabled
  - dpm.h, 344
- dpm\_sleep
  - dpm.h, 344
- dpm\_start
  - dpm.h, 344
- dpm\_status\_t, 73
  - alert, 75
  - app\_cbk, 75
  - attach, 76
  - attached\_dev, 76
  - bist\_cm2\_enabled, 76
  - bootup, 76
  - cbl\_dsc, 76
  - cbl\_mode\_en, 76
  - cbl\_soft\_reset\_tried, 76
  - cbl\_state, 76
  - cbl\_type, 76
  - cbl\_vdm\_version, 77
  - cbl\_vdo, 77
  - cbl\_vdo\_2, 77
  - cbl\_wait, 77
  - cc\_live, 77
  - cc\_old\_status, 77
  - cc\_rd\_status, 77
  - cc\_status, 77

cmd\_p, 77  
 connect, 78  
 contract, 78  
 contract\_exist, 78  
 cur\_fb, 78  
 cur\_port\_role, 78  
 cur\_port\_type, 78  
 cur\_snk\_max\_min, 78  
 cur\_snk\_pdo, 78  
 cur\_snk\_pdo\_count, 78  
 cur\_src\_pdo, 79  
 cur\_src\_pdo\_count, 79  
 db\_support, 79  
 dead\_bat, 79  
 dflt\_port\_role, 79  
 dpm\_cmd\_buf, 79  
 dpm\_enabled, 79  
 dpm\_err\_info, 79  
 dpm\_init, 79  
 dpm\_pd\_cbk, 80  
 dpm\_pd\_cmd, 80  
 dpm\_pd\_cmd\_active, 80  
 dpm\_safe\_disable, 80  
 dpm\_typec\_cbk, 80  
 dpm\_typec\_cmd, 80  
 dpm\_typec\_cmd\_active, 80  
 drp\_period, 80  
 emca\_present, 80  
 err\_recov, 81  
 ext\_snk\_cap, 81  
 ext\_src\_cap, 81  
 fault\_active, 81  
 fr\_rx\_disabled, 81  
 fr\_rx\_en\_live, 81  
 fr\_tx\_disabled, 81  
 fr\_tx\_en\_live, 81  
 frs\_enable, 81  
 frs\_rx\_en, 82  
 frs\_tx\_en, 82  
 hw\_drp\_toggle\_en, 82  
 is\_snk\_bat, 82  
 is\_src\_bat, 82  
 non\_intr\_response, 82  
 padval, 82  
 pd3\_src\_cc\_busy, 82  
 pd\_connected, 82  
 pd\_disabled, 83  
 pd\_support, 83  
 pe\_evt, 83  
 pe\_fsm\_state, 83  
 polarity, 83  
 port\_disable, 83  
 port\_role, 83  
 port\_status, 83  
 pps\_src\_en, 83  
 pps\_status, 84  
 pwr\_limited, 84  
 ra\_present, 84  
 rand\_base, 84  
 reserved\_3, 84  
 rev3\_en, 84  
 rev\_pol, 84  
 role\_at\_connect, 84  
 rp\_supported, 84  
 skip\_scan, 85  
 snk\_cur\_level, 85  
 snk\_max\_min, 85  
 snk\_pdo, 85  
 snk\_pdo\_count, 85  
 snk\_pdo\_mask, 85  
 snk\_period, 85  
 snk\_rdo, 85  
 snk\_rp\_detach\_en, 86  
 snk\_sel\_pdo, 86  
 snk\_usb\_comm\_en, 86  
 snk\_usb\_susp\_en, 86  
 spec\_rev\_cbl, 86  
 spec\_rev\_peer, 86  
 spec\_rev\_sop\_live, 86  
 spec\_rev\_sop\_prime\_live, 86  
 src\_cap\_p, 86  
 src\_cap\_start\_delay, 87  
 src\_cur\_level, 87  
 src\_cur\_level\_live, 87  
 src\_cur\_rdo, 87  
 src\_last\_rdo, 87  
 src\_pdo, 87  
 src\_pdo\_count, 87  
 src\_pdo\_mask, 87  
 src\_period, 87  
 src\_rdo, 88  
 src\_sel\_pdo, 88  
 swap\_response, 88  
 toggle, 88  
 try\_src\_snk, 88  
 try\_src\_snk\_dis, 88  
 typec\_evt, 88  
 typec\_fsm\_state, 88  
 unchunk\_sup\_live, 89  
 unchunk\_sup\_peer, 89  
 usb4\_en, 89  
 vconn\_logical, 89  
 vconn\_retain, 89  
 dpm\_stop  
     dpm.h, 346  
 dpm\_task  
     dpm.h, 346  
 dpm\_typec\_cbk  
     dpm\_status\_t, 80  
 dpm\_typec\_cmd  
     dpm\_status\_t, 80  
 dpm\_typec\_cmd\_active  
     dpm\_status\_t, 80  
 dpm\_typec\_cmd\_cbk\_t  
     pd.h, 383  
 dpm\_typec\_cmd\_resp\_t

- pd.h, 390
- dpm\_typec\_cmd\_t
  - pd.h, 391
- dpm\_typec\_command
  - dpm.h, 346
- dpm\_typec\_deassert\_rp\_rd
  - dpm.h, 347
- dpm\_update\_def\_cable\_cap
  - dpm.h, 347
- dpm\_update\_ext\_snk\_cap
  - dpm.h, 347
- dpm\_update\_ext\_src\_cap
  - dpm.h, 348
- dpm\_update\_frs\_enable
  - dpm.h, 348
- dpm\_update\_mux\_enable\_wait\_period
  - dpm.h, 349
- dpm\_update\_ndiscover\_identity\_count
  - dpm.h, 349
- dpm\_update\_port\_config
  - dpm.h, 349
- dpm\_update\_port\_status
  - dpm.h, 350
- dpm\_update\_rp\_audio\_accessory
  - dpm.h, 350
- dpm\_update\_snk\_cap
  - dpm.h, 351
- dpm\_update\_snk\_cap\_mask
  - dpm.h, 351
- dpm\_update\_snk\_max\_min
  - dpm.h, 351
- dpm\_update\_snk\_wait\_cap\_period
  - dpm.h, 352
- dpm\_update\_src\_cap
  - dpm.h, 352
- dpm\_update\_src\_cap\_mask
  - dpm.h, 352
- dpm\_update\_swap\_response
  - dpm.h, 353
- dpm\_wakeup
  - dpm.h, 353
- dr\_swap
  - pd\_do\_t::FIXED\_SNK, 93
  - pd\_do\_t::FIXED\_SRC, 94
- drp\_period
  - dpm\_status\_t, 80
- drp\_toggle\_en
  - pd\_port\_config\_t, 127
- dual\_role\_power
  - pd\_do\_t::FIXED\_SNK, 93
  - pd\_do\_t::FIXED\_SRC, 95
- dummy
  - pd\_power\_status\_t, 137
- EMPTY\_VDO
  - alt\_modes\_mngr.h, 204
- EN\_FLAG\_MASK
  - alt\_modes\_mngr.h, 204
- EXIT\_ALL\_MODES
  - alt\_modes\_mngr.h, 205
- EXPECTED\_GOOD\_CRC\_CLEAR\_MASK
  - pdss\_hal.h, 444
- EXPECTED\_GOOD\_CRC\_HDR\_DIFF\_MASK\_REV3
  - pdss\_hal.h, 444
- EXPECTED\_GOOD\_CRC\_HDR\_MASK
  - pdss\_hal.h, 444
- emca\_present
  - dpm\_status\_t, 80
  - pd\_port\_status\_t::PD\_PORT\_STAT, 135
- emca\_spec\_rev
  - pd\_port\_status\_t::PD\_PORT\_STAT, 135
- emca\_type
  - pd\_port\_status\_t::PD\_PORT\_STAT, 135
- en
  - pd\_do\_t::DP\_STATUS\_VDO, 70
- enable\_vdm\_task\_mngr
  - vdm\_task\_mngr.h, 229
- enter\_usb4
  - vdm\_task\_mngr.h, 229
- enterusb\_vdo
  - pd\_do\_t, 114
- err\_recov
  - dpm\_status\_t, 81
- err\_recovery\_en
  - pd\_port\_config\_t, 127
- eval\_app\_alt\_hw\_cmd
  - alt\_mode\_hw.h, 199
- eval\_app\_alt\_mode\_cmd
  - alt\_modes\_mngr.h, 211
- eval\_app\_cmd
  - alt\_mode\_info\_t, 35
- eval\_dr\_swap
  - app\_cbk\_t, 39
  - swap.h, 310
- eval\_enter\_usb
  - app\_cbk\_t, 39
  - vdm.h, 311
- eval\_fr\_swap
  - app\_cbk\_t, 39
  - swap.h, 310
- eval\_hpd\_cmd
  - alt\_mode\_hw.h, 200
- eval\_mux\_cmd
  - alt\_mode\_hw.h, 200
- eval\_pr\_swap
  - app\_cbk\_t, 39
  - swap.h, 310
- eval\_rdo
  - app\_cbk\_t, 39
  - pdo.h, 304
- eval\_rec\_vdm
  - alt\_modes\_mngr.h, 211
- eval\_src\_cap
  - app\_cbk\_t, 39
  - pdo.h, 304
- eval\_vconn\_swap
  - app\_cbk\_t, 39

- swap.h, 311
- eval\_vdm
  - app\_cbk\_t, 40
  - vdm.h, 312
- event\_flags
  - pd\_power\_status\_t, 138
- event\_group\_clear\_event
  - utils.h, 572
- event\_group\_clear\_events\_by\_val
  - utils.h, 572
- event\_group\_get\_event
  - utils.h, 572
- event\_group\_set\_event
  - utils.h, 573
- event\_group\_set\_events\_by\_val
  - utils.h, 573
- evt\_data
  - alt\_mode\_evt\_t::ALT\_MODE\_EVT\_DATA, 31
  - alt\_mode\_hw\_evt\_t::ALT\_MODE\_HW\_EVT, 33
- evt\_type
  - alt\_mode\_evt\_t::ALT\_MODE\_EVT\_DATA, 31
- exit
  - pd\_do\_t::DP\_STATUS\_VDO, 70
- ext\_powered
  - pd\_do\_t::FIXED\_SNK, 93
  - pd\_do\_t::FIXED\_SRC, 95
- ext\_powered\_prs
  - custom\_host\_cfg\_settings\_t, 67
- ext\_snk\_cap
  - dpm\_status\_t, 81
- ext\_snk\_cap\_length
  - pd\_port\_config\_t, 127
- ext\_snk\_cap\_offset
  - pd\_port\_config\_t, 128
- ext\_src\_cap
  - dpm\_status\_t, 81
- ext\_src\_cap\_length
  - pd\_port\_config\_t, 128
- ext\_src\_cap\_offset
  - pd\_port\_config\_t, 128
- extd
  - pd\_extd\_hdr\_t, 118
  - pd\_hdr\_t::PD\_HDR, 120
- extd\_hdr
  - dpm\_pd\_cmd\_buf\_t, 72
- extd\_msg\_t
  - pd.h, 391
- extd\_type
  - dpm\_pd\_cmd\_buf\_t, 72
- FULL\_MASK
  - alt\_modes\_mngr.h, 205
- FW\_BASE\_VERSION
  - cgx\_version.h, 525
- fail\_status\_t
  - alt\_modes\_mngr.h, 208
- fault\_active
  - dpm\_status\_t, 81
- fault\_event\_handler
  - app.h, 254
- fault\_handler\_clear\_counts
  - app.h, 254
- fault\_handler\_init\_vars
  - app.h, 255
- fault\_handler\_task
  - app.h, 255
- fault\_status
  - app\_status\_t, 45
- fb\_ctrl\_r1
  - pwr\_params\_t, 147
- fb\_ctrl\_r2
  - pwr\_params\_t, 147
- fb\_type
  - pwr\_params\_t, 148
- filter\_edge\_detect\_cfg\_t
  - pdss\_hal.h, 451
- filter\_id\_t
  - pdss\_hal.h, 451
- first\_msg\_rcvd
  - pri\_cntrs\_t, 146
- fixed\_bats
  - pd\_do\_t::ADO\_ALERT, 28
- fixed\_snk
  - pd\_do\_t, 114
- fixed\_src
  - pd\_do\_t, 114
- flash.h
  - flash\_access\_get\_status, 528
  - flash\_app\_priority\_t, 527
  - flash\_cbk\_t, 527
  - flash\_enter\_mode, 528
  - flash\_get\_access\_limits, 529
  - flash\_interface\_t, 527
  - flash\_row\_clear, 529
  - flash\_row\_read, 530
  - flash\_row\_write, 530
  - flash\_set\_access\_limits, 531
  - flash\_set\_app\_priority, 531
  - flash\_write\_status\_t, 528
  - SROM\_API\_PARAM\_SIZE, 527
  - sflash\_row\_read, 532
  - sflash\_row\_write, 532
- flash\_access\_get\_status
  - flash.h, 528
- flash\_app\_priority\_t
  - flash.h, 527
- flash\_cbk\_t
  - flash.h, 527
- flash\_checksum
  - pd\_config\_t, 109
- flash\_enter\_mode
  - flash.h, 528
- flash\_get\_access\_limits
  - flash.h, 529
- flash\_interface\_t
  - flash.h, 527
- flash\_row\_clear

- flash.h, 529
- flash\_row\_read
  - flash.h, 530
- flash\_row\_write
  - flash.h, 530
- flash\_set\_access\_limits
  - flash.h, 531
- flash\_set\_app\_priority
  - flash.h, 531
- flash\_write\_status\_t
  - flash.h, 528
- flashing\_mode
  - pd\_config\_t, 109
- flashing\_vid
  - pd\_config\_t, 109
- form\_alt\_mode\_event
  - alt\_modes\_mngr.h, 211
- fr\_rx\_disabled
  - dpm\_status\_t, 81
- fr\_rx\_en\_live
  - dpm\_status\_t, 81
- fr\_swap
  - pd\_do\_t::FIXED\_SNK, 93
- fr\_swap\_supp\_t
  - pd.h, 391
- fr\_tx\_disabled
  - dpm\_status\_t, 81
- fr\_tx\_en\_live
  - dpm\_status\_t, 81
- frs\_enable
  - dpm\_status\_t, 81
  - pd\_port\_config\_t, 128
- frs\_rx
  - pd\_stack\_conf\_t, 139
- frs\_rx\_en
  - dpm\_status\_t, 82
- frs\_rx\_enable
  - pd\_status\_t, 141
- frs\_tx
  - pd\_stack\_conf\_t, 139
- frs\_tx\_en
  - dpm\_status\_t, 82
- frs\_tx\_enable
  - pd\_status\_t, 141
- frs\_tx\_source\_t
  - pdss\_hal.h, 452
- fw1\_invalid
  - fw\_img\_status\_t::fw\_mode\_reason\_t, 97
- fw2\_invalid
  - fw\_img\_status\_t::fw\_mode\_reason\_t, 97
- fw\_checksum
  - sys\_fw\_metadata\_t, 179
- fw\_entry
  - sys\_fw\_metadata\_t, 179
- fw\_img\_status\_t, 96
  - status, 96
  - val, 96
- fw\_img\_status\_t::fw\_mode\_reason\_t, 97
  - boot\_mode\_request, 97
  - fw1\_invalid, 97
  - fw2\_invalid, 97
  - reserved, 97
  - reserved1, 97
- fw\_size
  - sys\_fw\_metadata\_t, 179
- fw\_version
  - sys\_fw\_metadata\_t, 180
- GET\_DR\_SWAP\_RESP
  - pd.h, 368
- GET\_HPDI\_IRQ\_STAT
  - dp\_sid.h, 222
- GET\_LEGACY\_TBT\_ADAPTER
  - intel\_vid.h, 294
- GET\_MAX
  - utils.h, 568
- GET\_MIN
  - utils.h, 568
- GET\_PR\_SWAP\_RESP
  - pd.h, 368
- GET\_VCONN\_SWAP\_RESP
  - pd.h, 368
- GIVE\_BACK\_MASK
  - pd.h, 369
- GPIO\_DM\_FIELD\_MASK
  - gpio.h, 534
- GPIO\_DM\_FIELD\_SIZE
  - gpio.h, 534
- GPIO\_INT\_FIELD\_MASK
  - gpio.h, 534
- GPIO\_PORT0\_INTR\_NO
  - gpio.h, 534
- GPIO\_PORT1\_INTR\_NO
  - gpio.h, 534
- GPIO\_PORT2\_INTR\_NO
  - gpio.h, 534
- GPIO\_PORT3\_INTR\_NO
  - gpio.h, 534
- GPIO\_PORT4\_INTR\_NO
  - gpio.h, 534
- GPIO\_PORT5\_INTR\_NO
  - gpio.h, 535
- GPIO\_PORT6\_INTR\_NO
  - gpio.h, 535
- get\_alt\_mode\_numb
  - alt\_modes\_mngr.h, 212
- get\_alt\_modes\_config\_svid\_idx
  - alt\_modes\_mngr.h, 212
- get\_aux1\_resistor\_config
  - pdss\_hal.h, 458
- get\_aux2\_resistor\_config
  - pdss\_hal.h, 460
- get\_boot\_mode\_reason
  - boot.h, 524
- get\_custom\_svid
  - alt\_modes\_mngr.h, 213
- get\_mode\_info

- alt\_modes\_mngr.h, 213
- get\_mux\_state
  - alt\_mode\_hw.h, 200
- get\_pd\_config
  - pd.h, 404
- get\_pd\_port\_config
  - pd.h, 405
- get\_pe\_state\_buf
  - pd\_policy\_engine.h, 412
- get\_sbu1\_switch\_state
  - pdss\_hal.h, 460
- get\_sbu2\_switch\_state
  - pdss\_hal.h, 460
- get\_silicon\_revision
  - system.h, 550
- get\_spec\_rev\_determined
  - pd\_policy\_engine.h, 412
- get\_svid\_from\_idx
  - alt\_modes\_mngr.h, 213
- get\_vdm\_buff
  - alt\_modes\_mngr.h, 214
- give\_back\_flag
  - pd\_do\_t::RDO\_BAT\_GIVEBACK, 154
  - pd\_do\_t::RDO\_BAT, 152
  - pd\_do\_t::RDO\_FIXED\_VAR\_GIVEBACK, 157
  - pd\_do\_t::RDO\_FIXED\_VAR, 156
  - pd\_do\_t::RDO\_GEN\_GVB, 161
  - pd\_do\_t::RDO\_GEN, 159
- gl\_dpm\_port\_type
  - dpm.h, 353
- gl\_img1\_fw\_metadata
  - boot.h, 524
- gl\_img1\_fw\_pseudo\_metadata
  - boot.h, 524
- gl\_img2\_fw\_metadata
  - boot.h, 525
- gl\_img2\_fw\_pseudo\_metadata
  - boot.h, 525
- gl\_img\_status
  - boot.h, 525
- gpio.h
  - GPIO\_DM\_FIELD\_MASK, 534
  - GPIO\_DM\_FIELD\_SIZE, 534
  - GPIO\_INT\_FIELD\_MASK, 534
  - GPIO\_PORT0\_INTR\_NO, 534
  - GPIO\_PORT1\_INTR\_NO, 534
  - GPIO\_PORT2\_INTR\_NO, 534
  - GPIO\_PORT3\_INTR\_NO, 534
  - GPIO\_PORT4\_INTR\_NO, 534
  - GPIO\_PORT5\_INTR\_NO, 535
  - GPIO\_PORT6\_INTR\_NO, 535
  - gpio\_clear\_intr, 540
  - gpio\_dm\_t, 535
  - gpio\_get\_intr, 540
  - gpio\_hsiom\_set\_config, 540
  - gpio\_int\_set\_config, 541
  - gpio\_intr\_cb\_t, 535
  - gpio\_intr\_t, 537
  - gpio\_port\_pin\_t, 537
  - gpio\_read\_value, 541
  - gpio\_register\_intr\_cb, 542
  - gpio\_set\_drv\_mode, 542
  - gpio\_set\_lvttl\_mode, 542
  - gpio\_set\_value, 543
  - HSIOM\_FIELD\_SHIFT, 535
  - hsiom\_mode\_t, 539
  - hsiom\_set\_config, 543
  - gpio\_clear\_intr
    - gpio.h, 540
  - gpio\_dm\_t
    - gpio.h, 535
  - gpio\_get\_intr
    - gpio.h, 540
  - gpio\_hsiom\_set\_config
    - gpio.h, 540
  - gpio\_int\_set\_config
    - gpio.h, 541
  - gpio\_intr\_cb\_t
    - gpio.h, 535
  - gpio\_intr\_t
    - gpio.h, 537
  - gpio\_port\_pin\_t
    - gpio.h, 537
  - gpio\_read\_value
    - gpio.h, 541
  - gpio\_register\_intr\_cb
    - gpio.h, 542
  - gpio\_set\_drv\_mode
    - gpio.h, 542
  - gpio\_set\_lvttl\_mode
    - gpio.h, 542
  - gpio\_set\_value
    - gpio.h, 543
  - HOST\_SBU\_CFG\_FIXED\_POL
    - pd.h, 369
  - HOST\_SBU\_CFG\_FULL
    - pd.h, 369
  - HOST\_SBU\_CFG\_PASS\_THROUGH
    - pd.h, 369
  - HPD\_DISABLE\_CMD
    - alt\_mode\_hw.h, 196
  - HPD\_ENABLE\_CMD
    - alt\_mode\_hw.h, 196
  - HPD\_EVENT\_0\_POS
    - hpd.h, 433
  - HPD\_EVENT\_1\_POS
    - hpd.h, 433
  - HPD\_EVENT\_2\_POS
    - hpd.h, 433
  - HPD\_EVENT\_3\_POS
    - hpd.h, 433
  - HPD\_EVENT\_MASK
    - hpd.h, 433
  - HPD\_GET\_EVENT\_0
    - hpd.h, 433
  - HPD\_GET\_EVENT\_1



- hpd.h, 434
- HPD\_GET\_EVENT\_2
  - hpd.h, 434
- HPD\_GET\_EVENT\_3
  - hpd.h, 434
- HPD\_HIGH\_IRQ\_HIGH
  - dp\_sid.h, 222
- HPD\_HIGH\_IRQ\_LOW
  - dp\_sid.h, 223
- HPD\_IP\_DIRECTION
  - hpd.h, 434
- HPD\_IRQ\_BIT\_POS
  - dp\_sid.h, 223
- HPD\_LOW\_IRQ\_HIGH
  - dp\_sid.h, 223
- HPD\_LOW\_IRQ\_LOW
  - dp\_sid.h, 223
- HPD\_OP\_DIRECTION
  - hpd.h, 434
- HPD\_RX\_ACTIVITY\_TIMER\_PERIOD\_MAX
  - pd.h, 369
- HPD\_RX\_ACTIVITY\_TIMER\_PERIOD\_MIN
  - pd.h, 369
- HPD\_STATE\_BIT\_POS
  - dp\_sid.h, 223
- HPI\_ADDR\_I2C\_CFG\_FLOAT
  - hpi.h, 315
- HPI\_ADDR\_I2C\_CFG\_HIGH
  - hpi.h, 315
- HPI\_ADDR\_I2C\_CFG\_LOW
  - hpi.h, 315
- HPI\_AM\_SVID
  - custom\_hpi\_vid.h, 217
- HSIOM\_FIELD\_SHIFT
  - gpio.h, 535
- hal\_ccgx.h
  - system\_connect\_ovp\_trip, 428
  - system\_disconnect\_ovp\_trip, 428
  - system\_init, 429
  - system\_vbus\_ocp\_dis, 429
  - system\_vbus\_ocp\_en, 429
  - system\_vbus\_rcp\_dis, 429
  - system\_vbus\_rcp\_en, 430
  - system\_vbus\_scp\_dis, 430
  - system\_vbus\_scp\_en, 430
  - vbus\_ocp\_handler, 431
  - vbus\_rcp\_handler, 431
  - vbus\_scp\_handler, 431
- hdr
  - pd\_hdr\_t, 121
  - pd\_packet\_extd\_t, 122
  - pd\_packet\_t, 123
- high\_cap
  - pd\_do\_t::FIXED\_SNK, 93
- host\_cap
  - pd\_do\_t::DFP\_VDO, 68
- host\_dp\_supp
  - pd\_do\_t::ENTERUSB\_VDO, 90
- host\_pcie\_supp
  - pd\_do\_t::ENTERUSB\_VDO, 90
- host\_present
  - pd\_do\_t::ENTERUSB\_VDO, 90
- host\_support
  - tbthost\_cfg\_settings\_t, 185
- host\_tbt\_supp
  - pd\_do\_t::ENTERUSB\_VDO, 90
- hot\_swap\_bats
  - pd\_do\_t::ADO\_ALERT, 28
- hpd.h
  - HPD\_EVENT\_0\_POS, 433
  - HPD\_EVENT\_1\_POS, 433
  - HPD\_EVENT\_2\_POS, 433
  - HPD\_EVENT\_3\_POS, 433
  - HPD\_EVENT\_MASK, 433
  - HPD\_GET\_EVENT\_0, 433
  - HPD\_GET\_EVENT\_1, 434
  - HPD\_GET\_EVENT\_2, 434
  - HPD\_GET\_EVENT\_3, 434
  - HPD\_IP\_DIRECTION, 434
  - HPD\_OP\_DIRECTION, 434
  - hpd\_deinit, 435
  - hpd\_event\_type\_t, 434
  - hpd\_receive\_get\_status, 435
  - hpd\_receive\_init, 435
  - hpd\_rx\_sleep\_entry, 436
  - hpd\_rx\_wakeup, 436
  - hpd\_sleep\_entry, 436
  - hpd\_transmit\_init, 437
  - hpd\_transmit\_sendevt, 437
  - hpd\_wakeup, 438
  - is\_hpd\_rx\_state\_idle, 438
- hpd\_deinit
  - hpd.h, 435
- hpd\_event\_type\_t
  - hpd.h, 434
- hpd\_handling
  - tbthost\_cfg\_settings\_t, 185
- hpd\_irq
  - pd\_do\_t::DP\_STATUS\_VDO, 71
- hpd\_receive\_get\_status
  - hpd.h, 435
- hpd\_receive\_init
  - hpd.h, 435
- hpd\_rx\_sleep\_entry
  - hpd.h, 436
- hpd\_rx\_wakeup
  - hpd.h, 436
- hpd\_sleep\_entry
  - hpd.h, 436
- hpd\_state
  - pd\_do\_t::DP\_STATUS\_VDO, 71
- hpd\_transmit\_init
  - hpd.h, 437
- hpd\_transmit\_sendevt
  - hpd.h, 437
- hpd\_wakeup

- hpd.h, 438
- hpdt\_ctrl1\_reg\_check\_start
  - pdss\_hal.h, 461
- hpi.h
  - HPI\_ADDR\_I2C\_CFG\_FLOAT, 315
  - HPI\_ADDR\_I2C\_CFG\_HIGH, 315
  - HPI\_ADDR\_I2C\_CFG\_LOW, 315
  - hpi\_boot\_prio\_conf\_t, 316
  - hpi\_clear\_event, 317
  - hpi\_deinit, 318
  - hpi\_get\_port\_enable, 318
  - hpi\_get\_sys\_pwr\_state, 318
  - hpi\_get\_ucsi\_control, 318
  - hpi\_init, 319
  - hpi\_init\_userdef\_regs, 319
  - hpi\_is\_accessed, 319
  - hpi\_is\_ec\_ready, 320
  - hpi\_is\_extd\_msg\_ec\_ctrl\_enabled, 320
  - hpi\_is\_vdm\_ec\_ctrl\_enabled, 320
  - hpi\_pd\_event\_handler, 321
  - hpi\_reg\_enqueue\_event, 321
  - hpi\_reg\_part\_t, 316
  - hpi\_reg\_section\_t, 316
  - hpi\_send\_fw\_ready\_event, 322
  - hpi\_send\_hw\_error\_event, 322
  - hpi\_set\_boot\_priority\_conf, 322
  - hpi\_set\_ec\_interrupt, 322
  - hpi\_set\_event, 323
  - hpi\_set\_fixed\_slave\_address, 323
  - hpi\_set\_flash\_params, 323
  - hpi\_set\_hpi\_version, 324
  - hpi\_set\_mode\_regs, 324
  - hpi\_set\_no\_boot\_mode, 325
  - hpi\_set\_port\_event\_mask, 325
  - hpi\_set\_userdef\_write\_handler, 325
  - hpi\_sleep, 325
  - hpi\_sleep\_allowed, 326
  - hpi\_task, 326
  - hpi\_ucsi\_control\_cmds\_t, 317
  - hpi\_ucsi\_status\_reg\_t, 317
  - hpi\_update\_fw\_locations, 326
  - hpi\_update\_pdo\_change, 327
  - hpi\_update\_versions, 327
  - hpi\_write\_cb\_t, 315
  - hpid\_get\_ec\_active\_modes, 327
  - i2c\_owner\_t, 317
  - ucsi\_clear\_status\_bit, 328
  - ucsi\_get\_status\_bit, 328
  - ucsi\_reg\_reset, 328
  - ucsi\_set\_status\_bit, 328
- hpi\_am\_state\_t
  - custom\_hpi\_vid.h, 217
- hpi\_boot\_prio\_conf\_t
  - hpi.h, 316
- hpi\_clear\_event
  - hpi.h, 317
- hpi\_deinit
  - hpi.h, 318
- hpi\_get\_port\_enable
  - hpi.h, 318
- hpi\_get\_sys\_pwr\_state
  - hpi.h, 318
- hpi\_get\_ucsi\_control
  - hpi.h, 318
- hpi\_init
  - hpi.h, 319
- hpi\_init\_userdef\_regs
  - hpi.h, 319
- hpi\_is\_accessed
  - hpi.h, 319
- hpi\_is\_ec\_ready
  - hpi.h, 320
- hpi\_is\_extd\_msg\_ec\_ctrl\_enabled
  - hpi.h, 320
- hpi\_is\_vdm\_ec\_ctrl\_enabled
  - hpi.h, 320
- hpi\_pd\_event\_handler
  - hpi.h, 321
- hpi\_reg\_enqueue\_event
  - hpi.h, 321
- hpi\_reg\_part\_t
  - hpi.h, 316
- hpi\_reg\_section\_t
  - hpi.h, 316
- hpi\_send\_fw\_ready\_event
  - hpi.h, 322
- hpi\_send\_hw\_error\_event
  - hpi.h, 322
- hpi\_set\_boot\_priority\_conf
  - hpi.h, 322
- hpi\_set\_ec\_interrupt
  - hpi.h, 322
- hpi\_set\_event
  - hpi.h, 323
- hpi\_set\_fixed\_slave\_address
  - hpi.h, 323
- hpi\_set\_flash\_params
  - hpi.h, 323
- hpi\_set\_hpi\_version
  - hpi.h, 324
- hpi\_set\_mode\_regs
  - hpi.h, 324
- hpi\_set\_no\_boot\_mode
  - hpi.h, 325
- hpi\_set\_port\_event\_mask
  - hpi.h, 325
- hpi\_set\_userdef\_write\_handler
  - hpi.h, 325
- hpi\_sleep
  - hpi.h, 325
- hpi\_sleep\_allowed
  - hpi.h, 326
- hpi\_task
  - hpi.h, 326
- hpi\_ucsi\_control\_cmds\_t
  - hpi.h, 317

hpi\_ucsi\_status\_reg\_t  
   hpi.h, 317  
 hpi\_update\_fw\_locations  
   hpi.h, 326  
 hpi\_update\_pdo\_change  
   hpi.h, 327  
 hpi\_update\_versions  
   hpi.h, 327  
 hpi\_write\_cb\_t  
   hpi.h, 315  
 hpid\_get\_ec\_active\_modes  
   hpi.h, 327  
 hpiss/hpi.h, 313  
 hsiom\_mode\_t  
   gpio.h, 539  
 hsiom\_set\_config  
   gpio.h, 543  
 hw\_drp\_toggle\_en  
   dpm\_status\_t, 82  
 hw\_evt  
   alt\_mode\_hw\_evt\_t, 34  
 hw\_type  
   alt\_mode\_hw\_evt\_t::ALT\_MODE\_HW\_EVT, 33  
  
 I2C\_BLOCK\_COUNT  
   i2c.h, 512  
 I2C\_CLEAR\_INTR\_MASK  
   i2c.h, 513  
 I2C\_CLEAR\_INTR\_REQUEST\_REG  
   i2c.h, 513  
 I2C\_SCB\_FIFO\_SIZE  
   i2c.h, 513  
 I2C\_SCB\_RX\_FIFO\_SIZE  
   i2c.h, 513  
 I2C\_SCB\_TX\_FIFO\_SIZE  
   i2c.h, 513  
 I2C\_SLAVE\_ADDR\_MASK\_DEFAULT  
   i2c.h, 513  
 I2C\_SLAVE\_TIMER\_BASE  
   i2c.h, 513  
 I2C\_SLAVE\_TIMER\_PERIOD  
   i2c.h, 513  
 i2c.h  
   I2C\_BLOCK\_COUNT, 512  
   I2C\_CLEAR\_INTR\_MASK, 513  
   I2C\_CLEAR\_INTR\_REQUEST\_REG, 513  
   I2C\_SCB\_FIFO\_SIZE, 513  
   I2C\_SCB\_RX\_FIFO\_SIZE, 513  
   I2C\_SCB\_TX\_FIFO\_SIZE, 513  
   I2C\_SLAVE\_ADDR\_MASK\_DEFAULT, 513  
   I2C\_SLAVE\_TIMER\_BASE, 513  
   I2C\_SLAVE\_TIMER\_PERIOD, 513  
   i2c\_cb\_cmd\_t, 514  
   i2c\_reset, 515  
   i2c\_scb\_clock\_freq\_t, 514  
   i2c\_scb\_deinit, 515  
   i2c\_scb\_enable\_wakeup, 516  
   i2c\_scb\_init, 516  
   i2c\_scb\_is\_idle, 517  
   i2c\_scb\_mode\_t, 514  
   i2c\_scb\_state\_t, 514  
   i2c\_scb\_write, 517  
   i2c\_slave\_ack\_ctrl, 518  
   i2c\_timer\_cb, 518  
   i2c\_write\_count, 99  
   i2c\_scb\_config\_t, 98  
   buf\_size, 98  
   buffer, 98  
   cb\_fun\_ptr, 98  
   clock\_freq, 98  
   i2c\_state, 98  
   i2c\_write\_count, 99  
   mode, 99  
   slave\_address, 99  
   slave\_mask, 99  
  
 i2c\_scb\_deinit  
   i2c.h, 515  
 i2c\_scb\_enable\_wakeup  
   i2c.h, 516  
 i2c\_scb\_init  
   i2c.h, 516  
 i2c\_scb\_is\_idle  
   i2c.h, 517  
 i2c\_scb\_mode\_t  
   i2c.h, 514  
 i2c\_scb\_state\_t  
   i2c.h, 514  
 i2c\_scb\_write  
   i2c.h, 517  
 i2c\_slave\_ack\_ctrl  
   i2c.h, 518  
 i2c\_state  
   i2c\_scb\_config\_t, 98  
 i2c\_timer\_cb  
   i2c.h, 518  
 i2c\_write\_count  
   i2c\_scb\_config\_t, 99  
 I\_0P9A  
   pd.h, 369  
 I\_1P5A  
   pd.h, 369  
 I\_1A  
   pd.h, 369  
 I\_2A  
   pd.h, 369  
 I\_3A  
   pd.h, 370  
 I\_4A  
   pd.h, 370  
 I\_5A

- pd.h, 370
- ICL\_ADP\_ATTACH\_DEBOUNCE\_TIME
  - icl.h, 286
- ICL\_ADP\_DETACH\_DEBOUNCE\_TIME
  - icl.h, 286
- ICL\_CTRL\_CMD\_FORCE\_TBT\_MODE
  - icl.h, 287
- ICL\_CTRL\_CMD\_SOC\_TO\_EVT\_DISABLE
  - icl.h, 287
- ICL\_RFU\_EXIT\_DURATION
  - icl.h, 287
- ICL\_SOC\_ACK\_TIMEOUT\_PERIOD
  - ridge\_slave.h, 297
- ICL\_STS\_REG\_FORCE\_TBT\_MODE
  - icl.h, 287
- ICL\_VSYS\_STABLE\_WAIT\_TIME
  - app.h, 241
- ID\_HEADER\_IDX
  - pd.h, 370
- INTEL\_VID
  - intel\_vid.h, 294
- IS\_FLAG\_CHECKED
  - alt\_modes\_mngr.h, 205
- ISAFE\_0A
  - pd.h, 370
- ISAFE\_DEF
  - pd.h, 370
- icl.h
  - ADP\_EDGE\_NONE, 286
  - ADP\_NEGEDGE, 286
  - ADP\_POSEDGE, 286
  - ADP\_STATE\_INIT, 286
  - ICL\_ADP\_ATTACH\_DEBOUNCE\_TIME, 286
  - ICL\_ADP\_DETACH\_DEBOUNCE\_TIME, 286
  - ICL\_CTRL\_CMD\_FORCE\_TBT\_MODE, 287
  - ICL\_CTRL\_CMD\_SOC\_TO\_EVT\_DISABLE, 287
  - ICL\_RFU\_EXIT\_DURATION, 287
  - ICL\_STS\_REG\_FORCE\_TBT\_MODE, 287
  - icl\_evt\_t, 287
  - icl\_init, 288
  - icl\_reg\_t, 287
  - icl\_set\_evt, 288
  - icl\_sleep\_allowed, 288
  - icl\_task, 289
  - icl\_vsys\_is\_present, 289
- icl\_dual\_retimer\_enable
  - icl\_tgl\_cfg\_settings\_t, 100
- icl\_evt\_t
  - icl.h, 287
- icl\_i2c\_pmc\_address
  - icl\_tgl\_cfg\_settings\_t, 100
- icl\_i2c\_retimer\_address
  - icl\_tgl\_cfg\_settings\_t, 100
- icl\_init
  - icl.h, 288
- icl\_reg\_t
  - icl.h, 287
- icl\_set\_evt
  - icl.h, 288
- icl\_sleep\_allowed
  - icl.h, 288
- icl\_task
  - icl.h, 289
- icl\_tgl\_cfg\_settings\_t, 99
  - icl\_dual\_retimer\_enable, 100
  - icl\_i2c\_pmc\_address, 100
  - icl\_i2c\_retimer\_address, 100
  - icl\_tgl\_selection, 100
  - reserved0, 100
  - soc\_mux\_config\_delay, 100
  - soc\_mux\_init\_delay, 100
  - table\_len, 100
- icl\_tgl\_selection
  - icl\_tgl\_cfg\_settings\_t, 100
- icl\_tgl\_tbl\_offset
  - pd\_port\_config\_t, 128
- icl\_vsys\_is\_present
  - icl.h, 289
- id
  - ccg\_timer\_t, 62
- id\_vdm\_length
  - pd\_port\_config\_t, 128
- id\_vdm\_offset
  - pd\_port\_config\_t, 128
- ignore\_mux\_changes
  - alt\_mode\_hw.h, 201
- inst\_evt\_t
  - instrumentation.h, 544
- instrumentation.h
  - inst\_evt\_t, 544
  - instrumentation\_init, 545
  - instrumentation\_start, 545
- instrumentation\_init
  - instrumentation.h, 545
- instrumentation\_start
  - instrumentation.h, 545
- intel\_mode
  - pd\_do\_t::TBT\_CBL\_VDO, 181
  - pd\_do\_t::TBT\_UFP\_VDO, 182
  - pd\_do\_t::TBT\_VDO, 184
- intel\_pf\_type\_t
  - pd.h, 392
- intel\_ridge.h
  - ridge\_eval\_cmd, 290
  - ridge\_force\_status\_update, 290
  - ridge\_set\_mux, 290
  - ridge\_set\_vpro, 291
  - ridge\_update\_dr, 291
  - tr\_hpd\_deinit, 291
  - tr\_hpd\_init, 292
  - tr\_hpd\_sendevt, 292
  - tr\_is\_hpd\_change, 292
- intel\_vid.h
  - BB\_STATUS, 294
  - GET\_LEGACY\_TBT\_ADAPTER, 294
  - INTEL\_VID, 294

- MAX\_TBT\_VDO\_NUMB, 294
- reg\_intel\_modes, 296
- TBT\_ALT\_MODE\_ID, 294
- TBT\_EXIT, 294
- TBT\_VDO\_IDX, 294
- tbt\_cbl\_gen\_t, 295
- tbt\_cbl\_speed\_t, 295
- tbt\_state\_t, 295
- USB2\_ENABLE, 294
- VPRO\_ALT\_MODE\_ID, 294
- intl\_temperature
  - pd\_power\_status\_t, 138
- is\_active
  - alt\_mode\_info\_t, 35
- is\_alt\_mode\_mgr\_idle
  - alt\_modes\_mgr.h, 214
- is\_hot\_shutdown
  - app\_status\_t, 45
- is\_hpd\_rx\_state\_idle
  - hpd.h, 438
- is\_mux\_busy
  - app\_status\_t, 45
- is\_port\_connect\_changed
  - ucsi.h, 581
- is\_snk\_bat
  - dpm\_status\_t, 82
  - pd\_port\_config\_t, 128
- is\_src\_bat
  - dpm\_status\_t, 82
  - pd\_port\_config\_t, 128
- is\_svid\_supported
  - alt\_modes\_mgr.h, 214
- is\_ufp\_disc\_started
  - vdm\_task\_mgr.h, 229
- is\_vbus\_on
  - app\_status\_t, 45
- is\_vconn\_on
  - app\_status\_t, 46
- is\_vdm\_pending
  - app\_status\_t, 46
- is\_vdm\_task\_idle
  - vdm\_task\_mgr.h, 230
- keep\_vconn\_src
  - app\_status\_t, 46
- last\_rcvd\_sop
  - pd\_status\_t, 141
- last\_tx\_sop
  - pd\_status\_t, 141
- ld\_sw\_ctrl
  - app\_status\_t, 46
- len
  - pd\_hdr\_t::PD\_HDR, 120
  - pd\_packet\_extd\_t, 122
  - pd\_packet\_t, 123
- link\_training
  - pd\_do\_t::TBT\_CBL\_VDO, 181
  - pd\_do\_t::TBT\_VDO, 184
- lscsa\_app\_config\_t
  - pdss\_hal.h, 452
- MAKE\_DWORD\_FROM\_WORD
  - utils.h, 569
- MAKE\_DWORD
  - utils.h, 568
- MAKE\_WORD
  - utils.h, 569
- MAX\_CABLE\_SVID\_SUPP
  - vdm\_task\_mgr.h, 227
- MAX\_CBL\_DSC\_ID\_COUNT
  - pd.h, 370
- MAX\_DISC\_SVID\_COUNT
  - vdm\_task\_mgr.h, 227
- MAX\_DP\_VDO\_NUMB
  - dp\_sid.h, 223
- MAX\_DPM\_CMD\_RETRY\_COUNT
  - ucsi.h, 578
- MAX\_EXTD\_MSG\_LEGACY\_LEN
  - pd.h, 370
- MAX\_EXTD\_PKT\_SIZE
  - pd.h, 370
- MAX\_EXTD\_PKT\_WORDS
  - pd.h, 371
- MAX\_HARD\_RESET\_COUNT
  - pd.h, 371
- MAX\_HPI\_AM\_VDO\_NUMB
  - custom\_hpi\_vid.h, 217
- MAX\_MESSAGE\_ID
  - pd.h, 371
- MAX\_NO\_OF\_DO
  - pd.h, 371
- MAX\_NO\_OF\_PDO
  - pd.h, 371
- MAX\_NO\_OF\_RETIMERS
  - bb\_retimer.h, 278
- MAX\_NO\_OF\_VDO
  - pd.h, 371
- MAX\_PR\_SWAP\_WAIT\_COUNT
  - pd.h, 371
- MAX\_RETRY\_CNT
  - alt\_modes\_mgr.h, 205
- MAX\_SOP\_TYPES
  - pd.h, 371
- MAX\_SRC\_CAP\_COUNT
  - pd.h, 371
- MAX\_SUPP\_ALT\_MODES
  - alt\_modes\_mgr.h, 205
- MAX\_SVID\_VDO\_SUPP
  - vdm\_task\_mgr.h, 227
- MAX\_TBT\_VDO\_NUMB
  - intel\_vid.h, 294
- MEM\_CMP
  - utils.h, 569
- MEM\_COPY
  - utils.h, 569
- MEM\_SET
  - utils.h, 569

MODE\_NOT\_SUPPORTED  
     alt\_modes\_mngr.h, 205  
 max\_cur  
     pasc\_valley\_table\_t, 107  
     pd\_do\_t::PPS\_SRC, 145  
 max\_cur\_power  
     pd\_do\_t::SRC\_GEN, 166  
 max\_current  
     pd\_do\_t::FIXED\_SRC, 95  
     pd\_do\_t::VAR\_SRC, 191  
 max\_op\_current  
     pd\_do\_t::RDO\_FIXED\_VAR, 156  
 max\_op\_power  
     pd\_do\_t::RDO\_BAT, 153  
 max\_op\_temp  
     pd\_do\_t::ACT\_CBL\_VDO\_2, 26  
 max\_power  
     pd\_do\_t::BAT\_SRC, 54  
 max\_power\_cur  
     pd\_do\_t::RDO\_GEN\_GVB, 161  
 max\_pwm\_duty\_cycle  
     pwr\_params\_t, 148  
 max\_vbus\_volt  
     pd\_do\_t::ACT\_CBL\_VDO\_1, 24  
     pd\_do\_t::ACT\_CBL\_VDO, 22  
     pd\_do\_t::PAS\_CBL\_VDO, 106  
 max\_volt  
     contract\_t, 65  
     pd\_do\_t::PPS\_SNK, 143  
     pd\_do\_t::PPS\_SRC, 145  
 max\_voltage  
     pd\_do\_t::BAT\_SNK, 53  
     pd\_do\_t::BAT\_SRC, 54  
     pd\_do\_t::SRC\_GEN, 166  
     pd\_do\_t::VAR\_SNK, 191  
     pd\_do\_t::VAR\_SRC, 192  
 mem\_calculate\_byte\_checksum  
     utils.h, 573  
 mem\_calculate\_dword\_checksum  
     utils.h, 574  
 mem\_calculate\_word\_checksum  
     utils.h, 574  
 mem\_copy  
     utils.h, 575  
 mem\_copy\_word  
     utils.h, 575  
 mem\_set  
     utils.h, 575  
 metadata\_valid  
     sys\_fw\_metadata\_t, 180  
 mfg\_info\_length  
     pd\_config\_t, 109  
 mfg\_info\_offset  
     pd\_config\_t, 109  
 min\_max\_power\_cur  
     pd\_do\_t::RDO\_GEN, 159  
 min\_op\_current  
     pd\_do\_t::RDO\_FIXED\_VAR\_GIVEBACK, 157  
 min\_op\_power  
     pd\_do\_t::RDO\_BAT\_GIVEBACK, 154  
 min\_state  
     pd\_port\_status\_t::PD\_PORT\_STAT, 135  
 min\_volt  
     contract\_t, 65  
     pd\_do\_t::PPS\_SNK, 143  
     pd\_do\_t::PPS\_SRC, 145  
 min\_voltage  
     pd\_do\_t::BAT\_SNK, 53  
     pd\_do\_t::BAT\_SRC, 54  
     pd\_do\_t::SRC\_GEN, 166  
     pd\_do\_t::VAR\_SNK, 191  
     pd\_do\_t::VAR\_SRC, 192  
 mod\_support  
     pd\_do\_t::STD\_VDM\_ID\_HDR, 177  
 modal\_op\_support  
     vdm\_task\_mngr.h, 232  
 mode  
     i2c\_scb\_config\_t, 99  
     pd\_do\_t::BIST\_DO, 61  
 mode\_state  
     alt\_mode\_info\_t, 35  
 mode\_vdm\_length  
     pd\_port\_config\_t, 129  
 mode\_vdm\_offset  
     pd\_port\_config\_t, 129  
 msg  
     pd\_packet\_extd\_t, 122  
     pd\_packet\_t, 123  
 msg\_id  
     pd\_hdr\_t::PD\_HDR, 120  
 msg\_type  
     pd\_hdr\_t::PD\_HDR, 120  
 mult\_fun  
     pd\_do\_t::DP\_STATUS\_VDO, 71  
 mux\_ctrl\_bb\_enable  
     app.h, 255  
 mux\_ctrl\_init  
     app.h, 256  
 mux\_ctrl\_set\_cfg  
     app.h, 256  
 mux\_poll\_cbk  
     app\_status\_t, 46  
 mux\_poll\_fnc\_cbk\_t  
     app.h, 242  
 mux\_poll\_status\_t  
     alt\_mode\_hw.h, 196  
 mux\_select\_t  
     alt\_mode\_hw.h, 197  
 NO\_DATA  
     alt\_mode\_hw.h, 196  
     alt\_modes\_mngr.h, 205  
 NONE\_MODE\_MASK  
     alt\_modes\_mngr.h, 205  
 NONE\_VDO  
     alt\_modes\_mngr.h, 205  
 no\_of\_cmd\_do

- dpm\_pd\_cmd\_buf\_t, 72
- no\_resp
  - vdm\_resp\_t, 194
- no\_usb\_suspend
  - pd\_do\_t::RDO\_BAT\_GIVEBACK, 154
  - pd\_do\_t::RDO\_BAT, 153
  - pd\_do\_t::RDO\_FIXED\_VAR\_GIVEBACK, 158
  - pd\_do\_t::RDO\_FIXED\_VAR, 156
  - pd\_do\_t::RDO\_GEN\_GVB, 161
  - pd\_do\_t::RDO\_GEN, 159
  - pd\_do\_t::RDO\_PPS, 162
- non\_intr\_response
  - dpm\_status\_t, 82
- non\_tbt\_mux
  - tbt\_host\_cfg\_settings\_t, 185
- OTP\_DEBOUNCE\_PERIOD
  - app.h, 241
- obj\_pos
  - alt\_mode\_info\_t, 36
  - pd\_do\_t::RDO\_BAT\_GIVEBACK, 154
  - pd\_do\_t::RDO\_BAT, 153
  - pd\_do\_t::RDO\_FIXED\_VAR\_GIVEBACK, 158
  - pd\_do\_t::RDO\_FIXED\_VAR, 156
  - pd\_do\_t::RDO\_GEN\_GVB, 161
  - pd\_do\_t::RDO\_GEN, 159
  - pd\_do\_t::RDO\_PPS, 162
  - pd\_do\_t::STD\_VDM\_HDR, 176
- ocp
  - pd\_do\_t::ADO\_ALERT, 28
- ocp\_settings\_t, 101
  - cs\_res, 101
  - debounce, 101
  - debounce2, 101
  - retry\_cnt, 101
  - sense\_res, 101
  - table\_len, 101
  - threshold, 102
  - threshold2, 102
- ocp\_tbl\_offset
  - pd\_port\_config\_t, 129
- old\_dp\_dm\_status
  - bc\_status\_t, 60
- op\_cond\_change
  - pd\_do\_t::ADO\_ALERT, 29
- op\_cur
  - pd\_do\_t::PPS\_SNK, 143
  - pd\_do\_t::RDO\_PPS, 163
- op\_current
  - pd\_do\_t::FIXED\_SNK, 93
  - pd\_do\_t::RDO\_FIXED\_VAR\_GIVEBACK, 158
  - pd\_do\_t::RDO\_FIXED\_VAR, 156
  - pd\_do\_t::VAR\_SNK, 191
- op\_power
  - pd\_do\_t::BAT\_SNK, 54
  - pd\_do\_t::RDO\_BAT\_GIVEBACK, 154
  - pd\_do\_t::RDO\_BAT, 153
- op\_power\_cur
  - pd\_do\_t::RDO\_GEN\_GVB, 161
- pd\_do\_t::RDO\_GEN, 159
- opt\_isolated
  - pd\_do\_t::ACT\_CBL\_VDO\_2, 26
- otp
  - pd\_do\_t::ADO\_ALERT, 29
- otp\_settings\_t, 102
  - cutoff\_val, 102
  - cutoff\_val\_1, 102
  - debounce, 103
  - reserved\_1, 103
  - restart\_val, 103
  - restart\_val\_1, 103
  - table\_len, 103
  - therm\_1\_enable, 103
  - therm\_type, 103
  - therm\_type\_1, 103
- otp\_tbl\_offset
  - pd\_port\_config\_t, 129
- out\_volt
  - pd\_do\_t::RDO\_PPS, 163
- ovp
  - pd\_do\_t::ADO\_ALERT, 29
- ovp\_settings\_t, 104
  - debounce, 104
  - debounce\_ms, 104
  - retry\_cnt, 104
  - table\_len, 104
  - threshold, 104
- ovp\_tbl\_offset
  - pd\_port\_config\_t, 129
- PB\_DEBOUNCE\_PERIOD
  - app.h, 241
- PD\_ADC\_CB\_T
  - pdss\_hal.h, 447
- PD\_ADC\_ID\_T
  - pdss\_hal.h, 453
- PD\_ADC\_INPUT\_T
  - pdss\_hal.h, 453
- PD\_ADC\_INT\_T
  - pdss\_hal.h, 453
- PD\_ADC\_VREF\_T
  - pdss\_hal.h, 454
- PD\_BIST\_CONT\_MODE\_TIMER\_PERIOD
  - pd.h, 372
- PD\_CBL\_DELAY\_TIMER\_PERIOD
  - pd.h, 372
- PD\_CBL\_DSC\_ID\_START\_TIMER\_PERIOD
  - pd.h, 372
- PD\_CBL\_DSC\_ID\_TIMER\_PERIOD
  - pd.h, 372
- PD\_CBL\_POWER\_UP\_TIMER\_PERIOD
  - pd.h, 372
- PD\_CBL\_READY\_TIMER\_PERIOD
  - pd.h, 372
- PD\_COLLISION\_SRC\_COOL\_OFF\_TIMER\_PERIOD
  - pd.h, 372
- PD\_CUR\_PER\_UNIT
  - pd.h, 372

PD\_CURRENT\_PPS\_MULTIPLIER  
     pd.h, [372](#)  
 PD\_DATA\_RESET\_COMPLETION\_DELAY  
     pd.h, [373](#)  
 PD\_DATA\_RESET\_TIMEOUT\_PERIOD  
     pd.h, [373](#)  
 PD\_DATA\_RESET\_TIMER\_PERIOD  
     pd.h, [373](#)  
 PD\_DISC\_ID\_AMA\_VDO\_IDX  
     vdm\_task\_mngr.h, [227](#)  
 PD\_DPM\_RESP\_REC\_RESP\_PERIOD  
     pd.h, [373](#)  
 PD\_EXTERNALLY\_POWERED\_BIT\_POS  
     pd.h, [373](#)  
 PD\_HARD\_RESET\_TX\_TIMER\_PERIOD  
     pd.h, [373](#)  
 PD\_MAX\_SRC\_CAP\_TRY  
     pd.h, [373](#)  
 PD\_MSG\_HDR\_REV2\_IGNORE\_MASK  
     pdss\_hal.h, [444](#)  
 PD\_NO\_RESPONSE\_TIMER\_PERIOD  
     pd.h, [373](#)  
 PD\_PHY\_BUSY\_TIMER\_PERIOD  
     pd.h, [373](#)  
 PD\_PPS\_SRC\_TIMER\_PERIOD  
     pd.h, [374](#)  
 PD\_PS\_HARD\_RESET\_TIMER\_PERIOD  
     pd.h, [374](#)  
 PD\_PS\_SNK\_TRANSITION\_TIMER\_PERIOD  
     pd.h, [374](#)  
 PD\_PS\_SRC\_OFF\_TIMER\_PERIOD  
     pd.h, [374](#)  
 PD\_PS\_SRC\_ON\_TIMER\_PERIOD  
     pd.h, [374](#)  
 PD\_PS\_SRC\_TRANS\_TIMER\_PERIOD  
     pd.h, [374](#)  
 PD\_RECEIVER\_RESPONSE\_TIMER\_PERIOD  
     pd.h, [374](#)  
 PD\_SENDER\_RESPONSE\_TIMER\_PERIOD  
     pd.h, [374](#)  
 PD\_SINK\_TX\_TIMER\_PERIOD  
     pd.h, [374](#)  
 PD\_SINK\_VBUS\_TURN\_OFF\_TIMER\_PERIOD  
     pd.h, [375](#)  
 PD\_SINK\_VBUS\_TURN\_ON\_TIMER\_PERIOD  
     pd.h, [375](#)  
 PD\_SINK\_WAIT\_CAP\_TIMER\_PERIOD  
     pd.h, [375](#)  
 PD\_SOURCE\_TRANSITION\_TIMER\_PERIOD  
     pd.h, [375](#)  
 PD\_SRC\_CAP\_TIMER\_PERIOD  
     pd.h, [375](#)  
 PD\_SRC\_RECOVER\_TIMER\_PERIOD  
     pd.h, [375](#)  
 PD\_SVID\_ID\_HDR\_VDO\_START\_IDX  
     vdm\_task\_mngr.h, [227](#)  
 PD\_SWAP\_SRC\_START\_TIMER\_PERIOD  
     pd.h, [375](#)  
 PD\_UFP\_VCONN\_DISCHARGE\_DURATION  
     pd.h, [375](#)  
 PD\_VBUS\_TURN\_OFF\_TIMER\_PERIOD  
     pd.h, [375](#)  
 PD\_VBUS\_TURN\_ON\_TIMER\_PERIOD  
     pd.h, [376](#)  
 PD\_VCONN\_OFF\_TIMER\_PERIOD  
     pd.h, [376](#)  
 PD\_VCONN\_ON\_TIMER\_PERIOD  
     pd.h, [376](#)  
 PD\_VCONN\_REAPPLIED\_TIMER\_PERIOD  
     pd.h, [376](#)  
 PD\_VCONN\_SRC\_DISC\_TIMER\_PERIOD  
     pd.h, [376](#)  
 PD\_VCONN\_SWAP\_INITIATOR\_DELAY\_PERIOD  
     pd.h, [376](#)  
 PD\_VCONN\_SWAP\_INITIATOR\_TIMER\_PERIOD  
     pd.h, [376](#)  
 PD\_VCONN\_TURN\_ON\_TIMER\_PERIOD  
     pd.h, [376](#)  
 PD\_VDM\_ENTER\_MODE\_RESPONSE\_TIMER\_PERIOD  
     RIOD  
     pd.h, [376](#)  
 PD\_VDM\_EXIT\_MODE\_RESPONSE\_TIMER\_PERIOD  
     pd.h, [377](#)  
 PD\_VDM\_RESPONSE\_TIMER\_PERIOD  
     pd.h, [377](#)  
 PD\_VOLT\_PER\_UNIT\_PPS  
     pd.h, [377](#)  
 PD\_VOLT\_PER\_UNIT  
     pd.h, [377](#)  
 PDSS\_CC\_FILT\_CYCLES  
     pdss\_hal.h, [444](#)  
 PDSS\_DRP\_HIGH\_PERIOD\_MS  
     pdss\_hal.h, [445](#)  
 PDSS\_DRP\_HIGH\_PERIOD\_VAL  
     pdss\_hal.h, [445](#)  
 PDSS\_DRP\_TOGGLE\_PERIOD\_MS  
     pdss\_hal.h, [445](#)  
 PDSS\_DRP\_TOGGLE\_PERIOD\_VAL  
     pdss\_hal.h, [445](#)  
 PDSS\_MAX\_RX\_MEM\_HALF\_SIZE  
     pdss\_hal.h, [445](#)  
 PDSS\_MAX\_RX\_MEM\_SIZE  
     pdss\_hal.h, [445](#)  
 PDSS\_MAX\_TX\_MEM\_HALF\_SIZE  
     pdss\_hal.h, [445](#)  
 PDSS\_MAX\_TX\_MEM\_SIZE  
     pdss\_hal.h, [445](#)  
 PGDO\_PD\_ISINK\_TERMINAL\_VAL\_1  
     pdss\_hal.h, [446](#)  
 PGDO\_PD\_ISINK\_TERMINAL\_VAL  
     pdss\_hal.h, [445](#)  
 PMC\_SLAVE\_ADDR\_PORT0  
     ridge\_slave.h, [297](#)  
 PMC\_SLAVE\_ADDR\_PORT1  
     ridge\_slave.h, [297](#)  
 POM\_BC



- ucsi.h, [578](#)
- POM\_CUR\_LEVEL\_1\_5A
  - ucsi.h, [578](#)
- POM\_CUR\_LEVEL\_3A
  - ucsi.h, [579](#)
- POM\_CUR\_LEVEL\_DEF
  - ucsi.h, [579](#)
- POM\_NO\_CONSUMER
  - ucsi.h, [579](#)
- POM\_PD
  - ucsi.h, [579](#)
- PPS\_CF\_VBUS\_DECREMENT\_STEP
  - psource.h, [307](#)
- PRODUCT\_TYPE\_VDO\_1\_IDX
  - pd.h, [377](#)
- PRODUCT\_TYPE\_VDO\_2\_IDX
  - pd.h, [377](#)
- PRODUCT\_TYPE\_VDO\_3\_IDX
  - pd.h, [377](#)
- PRODUCT\_VDO\_IDX
  - pd.h, [377](#)
- padval
  - dpm\_status\_t, [82](#)
- pas\_cbl\_vdo
  - pd\_do\_t, [115](#)
- pasc\_mode\_t
  - pdss\_hal.h, [452](#)
- pasc\_ptdrv\_cont\_cbk\_t
  - pdss\_hal.h, [447](#)
- pasc\_valley\_cbk\_t
  - pdss\_hal.h, [447](#)
- pasc\_valley\_table\_t, [107](#)
  - max\_cur, [107](#)
  - reserved\_0, [107](#)
  - safe\_valley, [107](#)
  - table, [107](#)
  - v0, [108](#)
  - v1, [108](#)
  - v2, [108](#)
- pd.h
  - apdo\_t, [384](#)
  - app\_evt\_t, [384](#)
  - app\_fault\_mask\_t, [386](#)
  - app\_req\_status\_t, [386](#)
  - app\_resp\_cbk\_t, [382](#)
  - app\_swap\_resp\_t, [387](#)
  - BC\_SINK\_1\_2\_MODE\_ENABLE\_MASK, [363](#)
  - BC\_SINK\_APPLE\_MODE\_ENABLE\_MASK, [363](#)
  - BC\_SRC\_1\_2\_MODE\_ENABLE\_MASK, [363](#)
  - BC\_SRC\_AFC\_MODE\_ENABLE\_MASK, [363](#)
  - BC\_SRC\_APPLE\_MODE\_ENABLE\_MASK, [363](#)
  - BC\_SRC\_QC\_4\_0\_MODE\_ENABLE\_MASK, [363](#)
  - BC\_SRC\_QC\_MODE\_ENABLE\_MASK, [363](#)
  - BC\_SRC\_QC\_VER\_2\_CLASS\_A\_VAL, [364](#)
  - BC\_SRC\_QC\_VER\_2\_CLASS\_B\_VAL, [364](#)
  - BC\_SRC\_QC\_VER\_3\_CLASS\_A\_VAL, [364](#)
  - BC\_SRC\_QC\_VER\_3\_CLASS\_B\_VAL, [364](#)
  - BDO\_HDR\_IDX, [364](#)
  - bist\_mode\_t, [387](#)
  - CC\_CHANNEL\_1, [364](#)
  - CC\_CHANNEL\_2, [364](#)
  - CCG\_CC\_STAT\_DRP\_TOGGLE, [364](#)
  - CCG\_CC\_STAT\_RD\_PRESENT, [364](#)
  - CCG\_CC\_STAT\_RP\_PRESENT, [365](#)
  - CCG\_CC\_STAT\_VCONN\_ACTIVE, [365](#)
  - CCG\_CC\_STAT\_ZOPEN, [365](#)
  - CCG\_DPM\_ERROR\_NO\_VCONN, [365](#)
  - CCG\_DPM\_ERROR\_NONE, [365](#)
  - CCG\_FRS\_RX\_ENABLE\_MASK, [365](#)
  - CCG\_FRS\_TX\_ENABLE\_MASK, [365](#)
  - CCG\_PD\_EXT\_PPS\_STATUS\_SIZE, [365](#)
  - CCG\_PD\_EXT\_SNKCAP\_BUF\_SIZE, [365](#)
  - CCG\_PD\_EXT\_SNKCAP\_SIZE, [366](#)
  - CCG\_PD\_EXT\_SNKCAP\_VERS\_INDEX, [366](#)
  - CCG\_PD\_EXT\_SRCCAP\_INP\_INDEX, [366](#)
  - CCG\_PD\_EXT\_SRCCAP\_INP\_UNCONSTRAINED, [366](#)
  - CCG\_PD\_EXT\_SRCCAP\_PDP\_INDEX, [366](#)
  - CCG\_PD\_EXT\_SRCCAP\_SIZE, [366](#)
  - CCG\_PD\_EXT\_STATUS\_SIZE, [366](#)
  - CCG\_PD\_FIX\_SRC\_PDO\_MASK\_REV2, [366](#)
  - CCG\_PD\_FIX\_SRC\_PDO\_MASK\_REV3, [366](#)
  - CCG\_PD\_FLAG\_CONTRACT\_NEG\_ACTIVE, [367](#)
  - CCG\_PD\_FLAG\_EXPLICIT\_CONTRACT, [367](#)
  - CCG\_PD\_FLAG\_POWER\_SINK, [367](#)
  - CCG\_PD\_FLAG\_SRC\_READY, [367](#)
  - CCG\_USB4\_EUDO\_USB4\_EN\_MASK, [367](#)
  - CCG\_USB\_MODE\_USB2, [367](#)
  - CCG\_USB\_MODE\_USB3, [367](#)
  - CCG\_USB\_MODE\_USB4, [367](#)
  - CERT\_STAT\_IDX, [367](#)
  - CY\_VID, [368](#)
  - cbl\_term\_t, [387](#)
  - cbl\_type\_t, [387](#)
  - cbl\_vbus\_cur\_t, [388](#)
  - ctrl\_msg\_t, [388](#)
  - DP\_HPD\_TYPE\_GPIO, [368](#)
  - DP\_HPD\_TYPE\_VIRTUAL, [368](#)
  - DP\_SVID, [368](#)
  - DRP\_TOGGLE\_PERIOD, [368](#)
  - data\_msg\_t, [389](#)
  - data\_reset\_state\_t, [389](#)
  - dpm\_pd\_cmd\_cbk\_t, [382](#)
  - dpm\_pd\_cmd\_t, [390](#)
  - dpm\_typec\_cmd\_cbk\_t, [383](#)
  - dpm\_typec\_cmd\_resp\_t, [390](#)
  - dpm\_typec\_cmd\_t, [391](#)
  - extd\_msg\_t, [391](#)
  - fr\_swap\_supp\_t, [391](#)
  - GET\_DR\_SWAP\_RESP, [368](#)
  - GET\_PR\_SWAP\_RESP, [368](#)
  - GET\_VCONN\_SWAP\_RESP, [368](#)
  - GIVE\_BACK\_MASK, [369](#)
  - get\_pd\_config, [404](#)
  - get\_pd\_port\_config, [405](#)
  - HOST\_SBU\_CFG\_FIXED\_POL, [369](#)

- HOST\_SBU\_CFG\_FULL, 369  
 HOST\_SBU\_CFG\_PASS\_THROUGH, 369  
 HPD\_RX\_ACTIVITY\_TIMER\_PERIOD\_MAX, 369  
 HPD\_RX\_ACTIVITY\_TIMER\_PERIOD\_MIN, 369  
 I\_0P9A, 369  
 I\_1P5A, 369  
 I\_1A, 369  
 I\_2A, 369  
 I\_3A, 370  
 I\_4A, 370  
 I\_5A, 370  
 ID\_HEADER\_IDX, 370  
 ISAFE\_0A, 370  
 ISAFE\_DEF, 370  
 intel\_pf\_type\_t, 392  
 MAX\_CBL\_DSC\_ID\_COUNT, 370  
 MAX\_EXTD\_MSG\_LEGACY\_LEN, 370  
 MAX\_EXTD\_PKT\_SIZE, 370  
 MAX\_EXTD\_PKT\_WORDS, 371  
 MAX\_HARD\_RESET\_COUNT, 371  
 MAX\_MESSAGE\_ID, 371  
 MAX\_NO\_OF\_DO, 371  
 MAX\_NO\_OF\_PDO, 371  
 MAX\_NO\_OF\_VDO, 371  
 MAX\_PR\_SWAP\_WAIT\_COUNT, 371  
 MAX\_SOP\_TYPES, 371  
 MAX\_SRC\_CAP\_COUNT, 371  
 PD\_BIST\_CONT\_MODE\_TIMER\_PERIOD, 372  
 PD\_CBL\_DELAY\_TIMER\_PERIOD, 372  
 PD\_CBL\_DSC\_ID\_START\_TIMER\_PERIOD, 372  
 PD\_CBL\_DSC\_ID\_TIMER\_PERIOD, 372  
 PD\_CBL\_POWER\_UP\_TIMER\_PERIOD, 372  
 PD\_CBL\_READY\_TIMER\_PERIOD, 372  
 PD\_COLLISION\_SRC\_COOL\_OFF\_TIMER\_PERIOD, 372  
 PD\_CUR\_PER\_UNIT, 372  
 PD\_CURRENT\_PPS\_MULTIPLE, 372  
 PD\_DATA\_RESET\_COMPLETION\_DELAY, 373  
 PD\_DATA\_RESET\_TIMEOUT\_PERIOD, 373  
 PD\_DATA\_RESET\_TIMER\_PERIOD, 373  
 PD\_DPM\_RESP\_REC\_RESP\_PERIOD, 373  
 PD\_EXTERNALLY\_POWERED\_BIT\_POS, 373  
 PD\_HARD\_RESET\_TX\_TIMER\_PERIOD, 373  
 PD\_MAX\_SRC\_CAP\_TRY, 373  
 PD\_NO\_RESPONSE\_TIMER\_PERIOD, 373  
 PD\_PHY\_BUSY\_TIMER\_PERIOD, 373  
 PD\_PPS\_SRC\_TIMER\_PERIOD, 374  
 PD\_PS\_HARD\_RESET\_TIMER\_PERIOD, 374  
 PD\_PS\_SNK\_TRANSITION\_TIMER\_PERIOD, 374  
 PD\_PS\_SRC\_OFF\_TIMER\_PERIOD, 374  
 PD\_PS\_SRC\_ON\_TIMER\_PERIOD, 374  
 PD\_PS\_SRC\_TRANS\_TIMER\_PERIOD, 374  
 PD\_RECEIVER\_RESPONSE\_TIMER\_PERIOD, 374  
 PD\_SENDER\_RESPONSE\_TIMER\_PERIOD, 374  
 PD\_SINK\_TX\_TIMER\_PERIOD, 374  
 PD\_SINK\_VBUS\_TURN\_OFF\_TIMER\_PERIOD, 375  
 PD\_SINK\_VBUS\_TURN\_ON\_TIMER\_PERIOD, 375  
 PD\_SINK\_WAIT\_CAP\_TIMER\_PERIOD, 375  
 PD\_SOURCE\_TRANSITION\_TIMER\_PERIOD, 375  
 PD\_SRC\_CAP\_TIMER\_PERIOD, 375  
 PD\_SRC\_RECOVER\_TIMER\_PERIOD, 375  
 PD\_SWAP\_SRC\_START\_TIMER\_PERIOD, 375  
 PD\_UFP\_VCONN\_DISCHARGE\_DURATION, 375  
 PD\_VBUS\_TURN\_OFF\_TIMER\_PERIOD, 375  
 PD\_VBUS\_TURN\_ON\_TIMER\_PERIOD, 376  
 PD\_VCONN\_OFF\_TIMER\_PERIOD, 376  
 PD\_VCONN\_ON\_TIMER\_PERIOD, 376  
 PD\_VCONN\_REAPPLIED\_TIMER\_PERIOD, 376  
 PD\_VCONN\_SRC\_DISC\_TIMER\_PERIOD, 376  
 PD\_VCONN\_SWAP\_INITIATOR\_DELAY\_PERIOD, 376  
 PD\_VCONN\_SWAP\_INITIATOR\_TIMER\_PERIOD, 376  
 PD\_VCONN\_TURN\_ON\_TIMER\_PERIOD, 376  
 PD\_VDM\_ENTER\_MODE\_RESPONSE\_TIMER\_PERIOD, 376  
 PD\_VDM\_EXIT\_MODE\_RESPONSE\_TIMER\_PERIOD, 377  
 PD\_VDM\_RESPONSE\_TIMER\_PERIOD, 377  
 PD\_VOLT\_PER\_UNIT\_PPS, 377  
 PD\_VOLT\_PER\_UNIT, 377  
 PRODUCT\_TYPE\_VDO\_1\_IDX, 377  
 PRODUCT\_TYPE\_VDO\_2\_IDX, 377  
 PRODUCT\_TYPE\_VDO\_3\_IDX, 377  
 PRODUCT\_VDO\_IDX, 377  
 pd\_ams\_type, 392  
 pd\_cable\_reset\_reason\_t, 392  
 pd\_cbk\_t, 383  
 pd\_contract\_status\_t, 393  
 pd\_devtype\_t, 393  
 pd\_emca\_sr\_reason\_t, 393  
 pd\_err\_recov\_reason\_t, 394  
 pd\_get\_ptr\_auto\_cfg\_tbl, 405  
 pd\_get\_ptr\_bat\_chg\_tbl, 405  
 pd\_get\_ptr\_bb\_tbl, 406  
 pd\_get\_ptr\_chg\_cfg\_tbl, 406  
 pd\_get\_ptr\_host\_cfg\_tbl, 406  
 pd\_get\_ptr\_icl\_tgl\_cfg\_tbl, 407  
 pd\_get\_ptr\_ocp\_tbl, 407  
 pd\_get\_ptr\_otp\_tbl, 407  
 pd\_get\_ptr\_ovp\_tbl, 408  
 pd\_get\_ptr\_pwr\_tbl, 408  
 pd\_get\_ptr\_rcp\_tbl, 408  
 pd\_get\_ptr\_scp\_tbl, 409  
 pd\_get\_ptr\_tbsthost\_cfg\_tbl, 409  
 pd\_get\_ptr\_type\_a\_chg\_cfg\_tbl, 409  
 pd\_get\_ptr\_type\_a\_pwr\_tbl, 410  
 pd\_get\_ptr\_uvp\_tbl, 410  
 pd\_get\_ptr\_vconn\_ocp\_tbl, 411

- pd\_hard\_reset\_reason\_t, 394
- pd\_is\_msg, 411
- pd\_msg\_class\_t, 394
- pd\_rev\_t, 395
- pd\_soft\_reset\_reason\_t, 395
- pdo\_sel\_alg\_t, 395
- pdo\_t, 396
- pe\_cbl\_state\_t, 396
- pe\_fsm\_state\_t, 396
- peak\_cur\_cap\_t, 397
- port\_role\_t, 398
- port\_type\_t, 398
- pwr\_ready\_cbk\_t, 383
- RDO\_IDX, 377
- rd\_cc\_status\_t, 398
- rdo\_type\_t, 398
- resp\_status\_t, 399
- rp\_cc\_status\_t, 399
- rp\_term\_t, 399
- SNK\_DETACH\_VBUS\_POLL\_COUNT, 378
- SNK\_MIN\_MAX\_MASK, 378
- SRC\_DRP\_MIN\_DC, 378
- STD\_SVID, 378
- STD\_VDM\_VERSION\_IDX, 378
- STD\_VDM\_VERSION\_REV2, 378
- STD\_VDM\_VERSION\_REV3, 378
- STD\_VDM\_VERSION, 378
- sink\_discharge\_off\_cbk\_t, 383
- sop\_t, 400
- std\_vdm\_cmd\_t, 400
- std\_vdm\_cmd\_type\_t, 400
- std\_vdm\_prod\_t, 401
- std\_vdm\_ver\_t, 401
- TBT\_CTRLR\_ALPINE\_RIDGE\_DUAL, 378
- TBT\_CTRLR\_ALPINE\_RIDGE\_SGL, 379
- TBT\_CTRLR\_TITAN\_RIDGE\_DUAL, 379
- TBT\_CTRLR\_TITAN\_RIDGE\_SGL, 379
- TBT\_GEN\_3, 379
- TBT\_SVID, 379
- TYPEC\_ATTACH\_WAIT\_ENTRY\_DELAY\_PERIOD, 379
- TYPEC\_CC\_DEBOUNCE\_TIMER\_PERIOD, 379
- TYPEC\_DRP\_TRY\_TIMER\_PERIOD, 379
- TYPEC\_ERROR\_RECOVERY\_TIMER\_PERIOD, 379
- TYPEC\_FSM\_GENERIC, 380
- TYPEC\_FSM\_NONE, 380
- TYPEC\_PD3\_RPCHANGE\_DEBOUNCE\_PERIOD, 380
- TYPEC\_PD\_DEBOUNCE\_TIMER\_PERIOD, 380
- TYPEC\_RD\_DEBOUNCE\_TIMER\_PERIOD, 380
- TYPEC\_SRC\_DETACH\_DEBOUNCE\_PERIOD, 380
- TYPEC\_TRY\_TIMEOUT\_PERIOD, 380
- try\_src\_snk\_t, 401
- typec\_fsm\_state\_t, 402
- UFP\_NON\_PH\_ALT\_MODE\_SUPP\_MASK, 380
- UFP\_VDO\_1\_RECFG\_ALT\_MODE\_PARAM\_MASK, 380
- usb\_data\_sig\_t, 402
- usb\_dev\_cap\_t, 402
- usb\_host\_cap\_t, 403
- usb\_role\_t, 403
- usb\_sig\_supp\_t, 403
- VDM\_HEADER\_IDX, 381
- VSAFE\_0V\_HARD\_RESET, 381
- VSAFE\_0V\_PR\_SWAP\_SNK\_SRC, 381
- VSAFE\_0V, 381
- VSAFE\_12V, 381
- VSAFE\_13V, 381
- VSAFE\_15V, 381
- VSAFE\_19V, 381
- VSAFE\_20V, 381
- VSAFE\_3\_6V, 382
- VSAFE\_5V, 382
- VSAFE\_9V, 382
- vdm\_ams\_t, 404
- vdm\_resp\_cbk\_t, 384
- vdm\_type\_t, 404
- pd3\_src\_cc\_busy
  - dpm\_status\_t, 82
- pd\_adc\_calibrate
  - pdss\_hal.h, 461
- pd\_adc\_comparator\_ctrl
  - pdss\_hal.h, 461
- pd\_adc\_comparator\_sample
  - pdss\_hal.h, 462
- pd\_adc\_free\_run\_ctrl
  - pdss\_hal.h, 462
- pd\_adc\_get\_comparator\_status
  - pdss\_hal.h, 463
- pd\_adc\_get\_vbus\_voltage
  - pdss\_hal.h, 463
- pd\_adc\_init
  - pdss\_hal.h, 464
- pd\_adc\_level\_to\_volt
  - pdss\_hal.h, 464
- pd\_adc\_sample
  - pdss\_hal.h, 465
- pd\_adc\_select\_vref
  - pdss\_hal.h, 465
- pd\_adc\_volt\_to\_level
  - pdss\_hal.h, 465
- pd\_ams\_type
  - pd.h, 392
- pd\_cable\_reset\_reason\_t
  - pd.h, 392
- pd\_cbk\_t
  - pd.h, 383
- pd\_cf\_disable
  - pdss\_hal.h, 466
- pd\_cf\_enable
  - pdss\_hal.h, 466
- pd\_cf\_get\_status
  - pdss\_hal.h, 467

- pd\_cf\_mon\_disable
  - pdss\_hal.h, 467
- pd\_cf\_mon\_enable
  - pdss\_hal.h, 467
- pd\_cmp\_cbk\_t
  - pdss\_hal.h, 448
- pd\_cmp\_get\_status
  - pdss\_hal.h, 468
- pd\_common/dpm.h, 329
- pd\_common/pd.h, 354
- pd\_common/pd\_policy\_engine.h, 411
- pd\_common/pd\_protocol.h, 415
- pd\_common/typec\_manager.h, 424
- pd\_config\_t, 108
  - application, 109
  - cable\_disc\_cnt, 109
  - db\_event\_mask, 109
  - flash\_checksum, 109
  - flashing\_mode, 109
  - flashing\_vid, 109
  - mfg\_info\_length, 109
  - mfg\_info\_offset, 109
  - pd\_port\_cnt, 109
  - port\_conf, 110
  - reserved\_0, 110
  - reserved\_2, 110
  - table\_checksum, 110
  - table\_sign, 110
  - table\_size, 110
  - table\_type, 110
  - table\_version, 110
- pd\_connect\_vbus\_div\_to\_amux
  - pdss\_hal.h, 468
- pd\_connected
  - dpm\_status\_t, 82
- pd\_contract\_info\_t, 111
  - rdo, 111
  - status, 111
- pd\_contract\_status\_t
  - pd.h, 393
- pd\_devtype\_t
  - pd.h, 393
- pd\_disabled
  - dpm\_status\_t, 83
- pd\_disconnect\_ra
  - pdss\_hal.h, 468
- pd\_disconnect\_vbus\_div\_from\_amux
  - pdss\_hal.h, 469
- pd\_do\_t, 111
  - act\_cbl\_vdo, 113
  - act\_cbl\_vdo1, 113
  - act\_cbl\_vdo2, 113
  - ado\_alert, 113
  - bat\_snk, 114
  - bat\_src, 114
  - bist\_do, 114
  - dfp\_vdo, 114
  - dp\_cfg\_vdo, 114
  - dp\_stat\_vdo, 114
  - enterusb\_vdo, 114
  - fixed\_snk, 114
  - fixed\_src, 114
  - pas\_cbl\_vdo, 115
  - pps\_snk, 115
  - pps\_src, 115
  - qc\_4\_0\_data\_vdo, 115
  - rdo\_bat, 115
  - rdo\_bat\_gvb, 115
  - rdo\_fix\_var, 115
  - rdo\_fix\_var\_gvb, 115
  - rdo\_gen, 115
  - rdo\_gen\_gvb, 116
  - rdo\_pps, 116
  - src\_gen, 116
  - std\_ama\_vdo, 116
  - std\_ama\_vdo\_pd3, 116
  - std\_cbl\_vdo, 116
  - std\_cert\_vdo, 116
  - std\_dp\_vdo, 116
  - std\_id\_hdr, 116
  - std\_prod\_vdo, 117
  - std\_svid\_res, 117
  - std\_vdm\_hdr, 117
  - tbt\_cbl\_vdo, 117
  - tbt\_ufp\_vdo, 117
  - tbt\_vdo, 117
  - ufp\_vdo\_1, 117
  - ustd\_qc\_4\_0\_hdr, 117
  - ustd\_vdm\_hdr, 117
  - val, 118
  - var\_snk, 118
  - var\_src, 118
- pd\_do\_t::ACT\_CBL\_VDO\_1, 23
  - cbl\_fw\_ver, 24
  - cbl\_hw\_ver, 24
  - cbl\_latency, 24
  - cbl\_term, 24
  - max\_vbus\_volt, 24
  - rsvd1, 24
  - rsvd2, 24
  - sbu\_supp, 24
  - sbu\_type, 24
  - sop\_dp, 25
  - typec\_abc, 25
  - usb\_ss\_sup, 25
  - vbus\_cur, 25
  - vbus\_thru\_cbl, 25
  - vdo\_version, 25
- pd\_do\_t::ACT\_CBL\_VDO\_2, 25
  - active\_el, 26
  - max\_op\_temp, 26
  - opt\_isolated, 26
  - phy\_conn, 26
  - rsvd0, 26
  - rsvd1, 26
  - shutdown\_temp, 27

- ss\_lanes, [27](#)
- ss\_supp, [27](#)
- u3\_power, [27](#)
- u3\_u0\_trans, [27](#)
- usb2\_hub\_hops, [27](#)
- usb2\_supp, [27](#)
- usb4\_supp, [27](#)
- usb\_gen, [27](#)
- pd\_do\_t::ACT\_CBL\_VDO, [21](#)
  - cbl\_fw\_ver, [21](#)
  - cbl\_hw\_ver, [21](#)
  - cbl\_latency, [22](#)
  - cbl\_term, [22](#)
  - max\_vbus\_volt, [22](#)
  - rsvd1, [22](#)
  - rsvd2, [22](#)
  - sop\_dp, [22](#)
  - typec\_abc, [22](#)
  - typec\_plug, [22](#)
  - usb\_ss\_sup, [22](#)
  - vbus\_cur, [23](#)
  - vbus\_thru\_cbl, [23](#)
  - vdo\_version, [23](#)
- pd\_do\_t::ADO\_ALERT, [28](#)
  - bat\_status\_change, [28](#)
  - fixed\_bats, [28](#)
  - hot\_swap\_bats, [28](#)
  - ocp, [28](#)
  - op\_cond\_change, [29](#)
  - otp, [29](#)
  - ovp, [29](#)
  - rsvd1, [29](#)
  - rsvd2, [29](#)
  - src\_input\_change, [29](#)
- pd\_do\_t::BAT\_SNK, [53](#)
  - max\_voltage, [53](#)
  - min\_voltage, [53](#)
  - op\_power, [54](#)
  - supply\_type, [54](#)
- pd\_do\_t::BAT\_SRC, [54](#)
  - max\_power, [54](#)
  - max\_voltage, [54](#)
  - min\_voltage, [54](#)
  - supply\_type, [55](#)
- pd\_do\_t::BIST\_DO, [60](#)
  - mode, [61](#)
  - rsvd1, [61](#)
  - rsvd2, [61](#)
- pd\_do\_t::DFP\_VDO, [68](#)
  - host\_cap, [68](#)
  - port\_num, [68](#)
  - rsvd0, [68](#)
  - rsvd1, [68](#)
  - vdo\_version, [68](#)
- pd\_do\_t::DP\_CONFIG\_VDO, [69](#)
  - dfp\_asgmt, [69](#)
  - rsvd1, [69](#)
  - rsvd2, [69](#)
  - sel\_conf, [69](#)
  - sign, [69](#)
  - ufp\_asgmt, [70](#)
- pd\_do\_t::DP\_STATUS\_VDO, [70](#)
  - dfp\_ufp\_conn, [70](#)
  - en, [70](#)
  - exit, [70](#)
  - hpd\_irq, [71](#)
  - hpd\_state, [71](#)
  - mult\_fun, [71](#)
  - pwr\_low, [71](#)
  - rsvd, [71](#)
  - usb\_cfg, [71](#)
- pd\_do\_t::ENTERUSB\_VDO, [89](#)
  - cable\_current, [90](#)
  - cable\_speed, [90](#)
  - cable\_type, [90](#)
  - host\_dp\_supp, [90](#)
  - host\_pcie\_supp, [90](#)
  - host\_present, [90](#)
  - host\_tbt\_supp, [90](#)
  - rsvd0, [90](#)
  - rsvd1, [91](#)
  - rsvd2, [91](#)
  - rsvd3, [91](#)
  - usb3\_drd, [91](#)
  - usb4\_drd, [91](#)
  - usb\_mode, [91](#)
- pd\_do\_t::FIXED\_SNK, [92](#)
  - dr\_swap, [93](#)
  - dual\_role\_power, [93](#)
  - ext\_powered, [93](#)
  - fr\_swap, [93](#)
  - high\_cap, [93](#)
  - op\_current, [93](#)
  - rsrvd, [93](#)
  - supply\_type, [94](#)
  - usb\_comm\_cap, [94](#)
  - voltage, [94](#)
- pd\_do\_t::FIXED\_SRC, [94](#)
  - dr\_swap, [94](#)
  - dual\_role\_power, [95](#)
  - ext\_powered, [95](#)
  - max\_current, [95](#)
  - pk\_current, [95](#)
  - reserved, [95](#)
  - supply\_type, [95](#)
  - unchunk\_sup, [95](#)
  - usb\_comm\_cap, [95](#)
  - usb\_suspend\_sup, [95](#)
  - voltage, [96](#)
- pd\_do\_t::PAS\_CBL\_VDO, [105](#)
  - cbl\_fw\_ver, [105](#)
  - cbl\_hw\_ver, [105](#)
  - cbl\_latency, [105](#)
  - cbl\_term, [105](#)
  - max\_vbus\_volt, [106](#)
  - rsvd1, [106](#)

- rsvd2, 106
- rsvd3, 106
- typec\_abc, 106
- typec\_plug, 106
- usb\_ss\_sup, 106
- vbus\_cur, 106
- vdo\_version, 106
- pd\_do\_t::PPS\_SNK, 143
  - apdo\_type, 143
  - cur\_fb, 143
  - max\_volt, 143
  - min\_volt, 143
  - op\_cur, 143
  - rsvd1, 143
  - rsvd2, 144
  - rsvd3, 144
  - rsvd4, 144
  - supply\_type, 144
- pd\_do\_t::PPS\_SRC, 144
  - apdo\_type, 145
  - max\_cur, 145
  - max\_volt, 145
  - min\_volt, 145
  - pps\_pwr\_limited, 145
  - rsvd1, 145
  - rsvd2, 145
  - rsvd3, 145
  - supply\_type, 145
- pd\_do\_t::QC\_4\_0\_DATA\_VDO, 150
  - data\_0, 151
  - data\_1, 151
  - data\_2, 151
  - data\_3, 151
- pd\_do\_t::RDO\_BAT\_GIVEBACK, 154
  - cap\_mismatch, 154
  - give\_back\_flag, 154
  - min\_op\_power, 154
  - no\_usb\_suspend, 154
  - obj\_pos, 154
  - op\_power, 154
  - rsrvd1, 155
  - rsrvd2, 155
  - unchunk\_sup, 155
  - usb\_comm\_cap, 155
- pd\_do\_t::RDO\_BAT, 152
  - cap\_mismatch, 152
  - give\_back\_flag, 152
  - max\_op\_power, 153
  - no\_usb\_suspend, 153
  - obj\_pos, 153
  - op\_power, 153
  - rsrvd1, 153
  - rsrvd2, 153
  - unchunk\_sup, 153
  - usb\_comm\_cap, 153
- pd\_do\_t::RDO\_FIXED\_VAR\_GIVEBACK, 157
  - cap\_mismatch, 157
  - give\_back\_flag, 157
  - min\_op\_current, 157
  - no\_usb\_suspend, 158
  - obj\_pos, 158
  - op\_current, 158
  - rsrvd1, 158
  - rsrvd2, 158
  - unchunk\_sup, 158
  - usb\_comm\_cap, 158
- pd\_do\_t::RDO\_FIXED\_VAR, 155
  - cap\_mismatch, 156
  - give\_back\_flag, 156
  - max\_op\_current, 156
  - no\_usb\_suspend, 156
  - obj\_pos, 156
  - op\_current, 156
  - rsrvd1, 156
  - rsrvd2, 156
  - unchunk\_sup, 156
  - usb\_comm\_cap, 157
- pd\_do\_t::RDO\_GEN\_GVB, 160
  - cap\_mismatch, 161
  - give\_back\_flag, 161
  - max\_power\_cur, 161
  - no\_usb\_suspend, 161
  - obj\_pos, 161
  - op\_power\_cur, 161
  - rsrvd1, 161
  - rsrvd2, 161
  - unchunk\_sup, 161
  - usb\_comm\_cap, 162
- pd\_do\_t::RDO\_GEN, 158
  - cap\_mismatch, 159
  - give\_back\_flag, 159
  - min\_max\_power\_cur, 159
  - no\_usb\_suspend, 159
  - obj\_pos, 159
  - op\_power\_cur, 159
  - rsrvd1, 160
  - rsrvd2, 160
  - unchunk\_sup, 160
  - usb\_comm\_cap, 160
- pd\_do\_t::RDO\_PPS, 162
  - cap\_mismatch, 162
  - no\_usb\_suspend, 162
  - obj\_pos, 162
  - op\_cur, 163
  - out\_volt, 163
  - rsvd1, 163
  - rsvd2, 163
  - rsvd3, 163
  - rsvd4, 163
  - unchunk\_sup, 163
  - usb\_comm\_cap, 163
- pd\_do\_t::SRC\_GEN, 166
  - max\_cur\_power, 166
  - max\_voltage, 166
  - min\_voltage, 166
  - supply\_type, 167

- pd\_do\_t::STD\_AMA\_VDO\_PD3, 169
  - ama\_fw\_ver, 169
  - ama\_hw\_ver, 169
  - rsvd1, 169
  - usb\_ss\_sup, 169
  - vbus\_req, 169
  - vcon\_pwr, 169
  - vcon\_req, 170
  - vdo\_version, 170
- pd\_do\_t::STD\_AMA\_VDO, 167
  - ama\_fw\_ver, 167
  - ama\_hw\_ver, 167
  - rsvd1, 167
  - ssrx1, 168
  - ssrx2, 168
  - sstx1, 168
  - sstx2, 168
  - usb\_ss\_sup, 168
  - vbus\_req, 168
  - vcon\_pwr, 168
  - vcon\_req, 168
- pd\_do\_t::STD\_CBL\_VDO, 170
  - cbl\_fw\_ver, 170
  - cbl\_hw\_ver, 171
  - cbl\_latency, 171
  - cbl\_term, 171
  - rsvd1, 171
  - sop\_dp, 171
  - ssrx1, 171
  - ssrx2, 171
  - sstx1, 171
  - sstx2, 171
  - typec\_abc, 172
  - typec\_plug, 172
  - usb\_ss\_sup, 172
  - vbus\_cur, 172
  - vbus\_thru\_cbl, 172
- pd\_do\_t::STD\_CERT\_VDO, 172
  - usb\_xid, 173
- pd\_do\_t::STD\_DP\_VDO, 173
  - dfp\_d\_pin, 173
  - port\_cap, 173
  - recep, 173
  - rsvd, 173
  - signal, 174
  - ufp\_d\_pin, 174
  - usb2\_0, 174
- pd\_do\_t::STD\_PROD\_VDO, 174
  - bcd\_dev, 174
  - usb\_pid, 174
- pd\_do\_t::STD\_SVID\_RESP\_VDO, 175
  - svid\_n, 175
  - svid\_n1, 175
- pd\_do\_t::STD\_VDM\_HDR, 175
  - cmd, 176
  - cmd\_type, 176
  - obj\_pos, 176
  - rsvd1, 176
  - rsvd2, 176
  - st\_ver, 176
  - svid, 176
  - vdm\_type, 176
- pd\_do\_t::STD\_VDM\_ID\_HDR, 177
  - mod\_support, 177
  - prod\_type, 177
  - prod\_type\_dfp, 177
  - rsvd1, 177
  - usb\_dev, 178
  - usb\_host, 178
  - usb\_vid, 178
- pd\_do\_t::TBT\_CBL\_VDO, 180
  - b22\_retimer\_cbl, 181
  - cbl\_gen, 181
  - cbl\_speed, 181
  - cbl\_type, 181
  - intel\_mode, 181
  - link\_training, 181
  - rsvd1, 181
- pd\_do\_t::TBT\_UFP\_VDO, 181
  - adapter, 182
  - intel\_mode, 182
  - rsvd0, 182
  - rsvd1, 182
  - vpro\_supp, 182
- pd\_do\_t::TBT\_VDO, 182
  - adapter, 183
  - b22\_retimer\_cbl, 183
  - cable\_active, 183
  - cbl\_gen, 183
  - cbl\_speed, 183
  - cbl\_type, 183
  - intel\_mode, 184
  - link\_training, 184
  - rsvd1, 184
  - rsvd2, 184
  - vpro\_dock\_host, 184
- pd\_do\_t::UFP\_VDO\_1, 186
  - alt\_modes, 186
  - dev\_cap, 187
  - rsvd0, 187
  - rsvd1, 187
  - usb\_sig, 187
  - vdo\_version, 187
- pd\_do\_t::USTD\_QC\_4\_0\_HDR, 187
  - cmd\_0, 188
  - cmd\_1, 188
  - svid, 188
  - vdm\_type, 188
- pd\_do\_t::USTD\_VDM\_HDR, 188
  - cmd, 188
  - cmd\_type, 189
  - rsvd1, 189
  - seq\_num, 189
  - svid, 189
  - vdm\_type, 189
  - vdm\_ver, 189

pd\_do\_t::VAR\_SNK, 190  
     max\_voltage, 191  
     min\_voltage, 191  
     op\_current, 191  
     supply\_type, 191  
 pd\_do\_t::VAR\_SRC, 191  
     max\_current, 191  
     max\_voltage, 192  
     min\_voltage, 192  
     supply\_type, 192  
 pd\_emca\_sr\_reason\_t  
     pd.h, 393  
 pd\_enable\_vconn\_comp  
     pdss\_hal.h, 469  
 pd\_err\_recov\_reason\_t  
     pd.h, 394  
 pd\_extd\_hdr\_t, 118  
     extd, 118  
     val, 118  
 pd\_extd\_hdr\_t::EXTD\_HDR\_T, 91  
     chunk\_no, 92  
     chunked, 92  
     data\_size, 92  
     request, 92  
     rsvd1, 92  
 pd\_fet\_automode\_disable  
     pdss\_hal.h, 469  
 pd\_fet\_automode\_enable  
     pdss\_hal.h, 470  
 pd\_fet\_dr\_t  
     pdss\_hal.h, 454  
 pd\_frs\_rx\_disable  
     pdss\_hal.h, 470  
 pd\_frs\_rx\_enable  
     pdss\_hal.h, 470  
 pd\_frs\_tx\_disable  
     pdss\_hal.h, 471  
 pd\_frs\_tx\_enable  
     pdss\_hal.h, 471  
 pd\_get\_ptr\_auto\_cfg\_tbl  
     pd.h, 405  
 pd\_get\_ptr\_bat\_chg\_tbl  
     pd.h, 405  
 pd\_get\_ptr\_bb\_tbl  
     pd.h, 406  
 pd\_get\_ptr\_cfg\_sub\_tbl  
     app.h, 256  
 pd\_get\_ptr\_chg\_cfg\_tbl  
     pd.h, 406  
 pd\_get\_ptr\_host\_cfg\_tbl  
     pd.h, 406  
 pd\_get\_ptr\_icl\_tgl\_cfg\_tbl  
     pd.h, 407  
 pd\_get\_ptr\_ocp\_tbl  
     pd.h, 407  
 pd\_get\_ptr\_otp\_tbl  
     pd.h, 407  
 pd\_get\_ptr\_ovp\_tbl  
     pd.h, 408  
 pd\_get\_ptr\_pwr\_tbl  
     pd.h, 408  
 pd\_get\_ptr\_rcp\_tbl  
     pd.h, 408  
 pd\_get\_ptr\_scp\_tbl  
     pd.h, 409  
 pd\_get\_ptr\_tbthost\_cfg\_tbl  
     pd.h, 409  
 pd\_get\_ptr\_type\_a\_chg\_cfg\_tbl  
     pd.h, 409  
 pd\_get\_ptr\_type\_a\_pwr\_tbl  
     pd.h, 410  
 pd\_get\_ptr\_uvp\_tbl  
     pd.h, 410  
 pd\_get\_ptr\_vconn\_ocp\_tbl  
     pd.h, 411  
 pd\_get\_vbus\_adc\_level  
     pdss\_hal.h, 471  
 pd\_get\_vconn\_status  
     pdss\_hal.h, 472  
 pd\_hal/hal\_ccgx.h, 427  
 pd\_hal/hpd.h, 432  
 pd\_hal/pdss\_hal.h, 438  
 pd\_hal\_abort\_auto\_toggle  
     pdss\_hal.h, 472  
 pd\_hal\_cleanup  
     pdss\_hal.h, 472  
 pd\_hal\_config\_auto\_toggle  
     pdss\_hal.h, 473  
 pd\_hal\_dual\_fet\_config  
     pdss\_hal.h, 473  
 pd\_hal\_enable\_internal\_vbus\_mon  
     pdss\_hal.h, 474  
 pd\_hal\_get\_vbus\_csa\_rsense  
     pdss\_hal.h, 474  
 pd\_hal\_get\_vbus\_detach\_adc  
     pdss\_hal.h, 474  
 pd\_hal\_get\_vbus\_detach\_input  
     pdss\_hal.h, 474  
 pd\_hal\_init  
     pdss\_hal.h, 475  
 pd\_hal\_is\_auto\_toggle\_active  
     pdss\_hal.h, 475  
 pd\_hal\_is\_sink\_fet\_on  
     pdss\_hal.h, 475  
 pd\_hal\_measure\_ea\_volt  
     pdss\_hal.h, 475  
 pd\_hal\_measure\_line\_volt  
     pdss\_hal.h, 476  
 pd\_hal\_measure\_vbus  
     pdss\_hal.h, 476  
 pd\_hal\_measure\_vbus\_cur  
     pdss\_hal.h, 476  
 pd\_hal\_measure\_vbus\_in  
     pdss\_hal.h, 477  
 pd\_hal\_set\_cc\_ovp\_pending  
     pdss\_hal.h, 477



- pd\_hal\_set\_fet\_drive
  - pdss\_hal.h, [477](#)
- pd\_hal\_set\_reference
  - pdss\_hal.h, [478](#)
- pd\_hal\_set\_supply\_change\_evt\_cb
  - pdss\_hal.h, [478](#)
- pd\_hal\_set\_vbus\_csa\_rsense
  - pdss\_hal.h, [479](#)
- pd\_hal\_set\_vbus\_detach\_params
  - pdss\_hal.h, [479](#)
- pd\_hal\_set\_vbus\_mon\_divider
  - pdss\_hal.h, [479](#)
- pd\_hal\_typec\_sm\_restart
  - pdss\_hal.h, [480](#)
- pd\_hal\_vconn\_ocp\_disable
  - pdss\_hal.h, [480](#)
- pd\_hal\_vconn\_ocp\_enable
  - pdss\_hal.h, [480](#)
- pd\_hard\_reset\_reason\_t
  - pd.h, [394](#)
- pd\_hdr\_t, [121](#)
  - hdr, [121](#)
  - val, [121](#)
- pd\_hdr\_t::PD\_HDR, [119](#)
  - chunk\_no, [119](#)
  - chunked, [119](#)
  - data\_role, [119](#)
  - data\_size, [119](#)
  - extd, [120](#)
  - len, [120](#)
  - msg\_id, [120](#)
  - msg\_type, [120](#)
  - pwr\_role, [120](#)
  - request, [120](#)
  - rsvd1, [120](#)
  - spec\_rev, [120](#)
- pd\_internal\_cfet\_off
  - pdss\_hal.h, [481](#)
- pd\_internal\_cfet\_on
  - pdss\_hal.h, [481](#)
- pd\_internal\_cfet\_soft\_start\_off
  - pdss\_hal.h, [482](#)
- pd\_internal\_cfet\_soft\_start\_on
  - pdss\_hal.h, [482](#)
- pd\_internal\_pfet\_off
  - pdss\_hal.h, [482](#)
- pd\_internal\_pfet\_on
  - pdss\_hal.h, [483](#)
- pd\_internal\_pfet\_soft\_start\_off
  - pdss\_hal.h, [483](#)
- pd\_internal\_pfet\_soft\_start\_on
  - pdss\_hal.h, [483](#)
- pd\_internal\_vbus\_discharge\_off
  - pdss\_hal.h, [484](#)
- pd\_internal\_vbus\_discharge\_on
  - pdss\_hal.h, [484](#)
- pd\_internal\_vbus\_in\_discharge\_off
  - pdss\_hal.h, [484](#)
- pd\_internal\_vbus\_in\_discharge\_on
  - pdss\_hal.h, [485](#)
- pd\_internal\_vbus\_load\_change\_isr\_enable
  - pdss\_hal.h, [485](#)
- pd\_internal\_vbus\_ocp\_dis
  - pdss\_hal.h, [486](#)
- pd\_internal\_vbus\_ocp\_en
  - pdss\_hal.h, [486](#)
- pd\_internal\_vbus\_ovp\_dis
  - pdss\_hal.h, [486](#)
- pd\_internal\_vbus\_ovp\_en
  - pdss\_hal.h, [487](#)
- pd\_internal\_vbus\_rcp\_dis
  - pdss\_hal.h, [487](#)
- pd\_internal\_vbus\_rcp\_en
  - pdss\_hal.h, [488](#)
- pd\_internal\_vbus\_scp\_dis
  - pdss\_hal.h, [488](#)
- pd\_internal\_vbus\_scp\_en
  - pdss\_hal.h, [489](#)
- pd\_internal\_vbus\_uvp\_dis
  - pdss\_hal.h, [489](#)
- pd\_internal\_vbus\_uvp\_en
  - pdss\_hal.h, [489](#)
- pd\_is\_msg
  - pd.h, [411](#)
- pd\_is\_v5v\_supply\_on
  - pdss\_hal.h, [490](#)
- pd\_is\_vconn\_present
  - pdss\_hal.h, [490](#)
- pd\_lscsa\_calc\_cfg
  - pdss\_hal.h, [491](#)
- pd\_lscsa\_cfg
  - pdss\_hal.h, [491](#)
- pd\_msg\_class\_t
  - pd.h, [394](#)
- pd\_operation\_en
  - pd\_port\_config\_t, [129](#)
- pd\_packet\_extd\_t, [121](#)
  - dat, [122](#)
  - data\_role, [122](#)
  - hdr, [122](#)
  - len, [122](#)
  - msg, [122](#)
  - sop, [122](#)
- pd\_packet\_t, [122](#)
  - dat, [123](#)
  - data\_role, [123](#)
  - hdr, [123](#)
  - len, [123](#)
  - msg, [123](#)
  - sop, [123](#)
- pd\_pasc\_get\_valley
  - pdss\_hal.h, [491](#)
- pd\_pasc\_lp\_disable
  - pdss\_hal.h, [492](#)
- pd\_pasc\_lp\_ds\_allowed
  - pdss\_hal.h, [492](#)

pd\_pasc\_lp\_enable  
     pdss\_hal.h, 492  
 pd\_pasc\_lp\_is\_active  
     pdss\_hal.h, 493  
 pd\_pasc\_lp\_task  
     pdss\_hal.h, 493  
 pd\_pasc\_poll\_task  
     pdss\_hal.h, 493  
 pd\_pasc\_ptdrv\_cont\_det\_disable  
     pdss\_hal.h, 494  
 pd\_pasc\_ptdrv\_cont\_det\_enable  
     pdss\_hal.h, 494  
 pd\_pasc\_send\_stop\_signal  
     pdss\_hal.h, 494  
 pd\_pasc\_set\_valley  
     pdss\_hal.h, 495  
 pd\_pasc\_start  
     pdss\_hal.h, 495  
 pd\_pasc\_valley\_algo\_enable  
     pdss\_hal.h, 495  
 pd\_pasc\_vbus\_in\_set\_volt  
     pdss\_hal.h, 496  
 pd\_pasc\_vbus\_in\_set\_volt\_abort  
     pdss\_hal.h, 496  
 pd\_pasc\_vbus\_in\_set\_vsaf\_5V  
     pdss\_hal.h, 496  
 pd\_phy\_abort\_bist\_cm2  
     pdss\_hal.h, 497  
 pd\_phy\_abort\_tx\_msg  
     pdss\_hal.h, 497  
 pd\_phy\_cbk\_t  
     pdss\_hal.h, 448  
 pd\_phy\_deepsleep  
     pdss\_hal.h, 497  
 pd\_phy\_dis\_unchunked\_tx  
     pdss\_hal.h, 498  
 pd\_phy\_en\_unchunked\_tx  
     pdss\_hal.h, 498  
 pd\_phy\_evt\_t  
     pdss\_hal.h, 454  
 pd\_phy\_get\_rx\_packet  
     pdss\_hal.h, 498  
 pd\_phy\_init  
     pdss\_hal.h, 499  
 pd\_phy\_is\_busy  
     pdss\_hal.h, 499  
 pd\_phy\_load\_msg  
     pdss\_hal.h, 499  
 pd\_phy\_refresh\_roles  
     pdss\_hal.h, 500  
 pd\_phy\_reset\_rx\_tx\_sm  
     pdss\_hal.h, 500  
 pd\_phy\_send\_bist\_cm2  
     pdss\_hal.h, 501  
 pd\_phy\_send\_msg  
     pdss\_hal.h, 501  
 pd\_phy\_send\_reset  
     pdss\_hal.h, 501  
 pd\_phy\_wakeup  
     pdss\_hal.h, 502  
 pd\_policy\_engine.h  
     get\_pe\_state\_buf, 412  
     get\_spec\_rev\_determined, 412  
     pe\_clear\_hard\_reset\_count, 412  
     pe\_disabled, 413  
     pe\_fsm, 413  
     pe\_get\_pps\_status, 413  
     pe\_init, 414  
     pe\_is\_busy, 414  
     pe\_push\_to\_buf, 414  
     pe\_start, 414  
     pe\_stop, 415  
 pd\_port\_cnt  
     pd\_config\_t, 109  
 pd\_port\_config\_t, 124  
     alt\_mode\_tbl\_offset, 125  
     alt\_mode\_trigger, 125  
     auto\_cfg\_tbl\_offset, 125  
     bat\_chg\_tbl\_offset, 125  
     bb\_tbl\_offset, 126  
     cable\_disc\_en, 126  
     chg\_cfg\_tbl\_offset, 126  
     current\_level, 126  
     custom\_alt\_mode\_tbl\_offset, 126  
     custom\_host\_tbl\_offset, 126  
     dead\_bat\_support, 126  
     default\_role, 126  
     default\_sink\_pdo\_mask, 126  
     default\_src\_pdo\_mask, 127  
     dock\_cfg\_tbl\_offset, 127  
     dp\_config\_supported, 127  
     dp\_mux\_control, 127  
     dp\_oper, 127  
     dp\_pref\_mode, 127  
     drp\_toggle\_en, 127  
     err\_recovery\_en, 127  
     ext\_snk\_cap\_length, 127  
     ext\_snk\_cap\_offset, 128  
     ext\_src\_cap\_length, 128  
     ext\_src\_cap\_offset, 128  
     frs\_enable, 128  
     icl\_tgl\_tbl\_offset, 128  
     id\_vdm\_length, 128  
     id\_vdm\_offset, 128  
     is\_snk\_bat, 128  
     is\_src\_bat, 128  
     mode\_vdm\_length, 129  
     mode\_vdm\_offset, 129  
     ocp\_tbl\_offset, 129  
     otp\_tbl\_offset, 129  
     ovp\_tbl\_offset, 129  
     pd\_operation\_en, 129  
     port\_disable, 129  
     port\_role, 129  
     protection\_enable, 129  
     pwr\_tbl\_offset, 130

- ra\_timeout, 130
- rcp\_tbl\_offset, 130
- reserved\_0, 130
- reserved\_1, 130
- reserved\_10, 130
- reserved\_11, 130
- reserved\_3, 130
- reserved\_4, 130
- reserved\_6, 131
- reserved\_8, 131
- reserved\_9, 131
- rp\_supported, 131
- scp\_tbl\_offset, 131
- snk\_pdo\_cnt, 131
- snk\_pdo\_list, 131
- snk\_pdo\_max\_min\_current\_pwr, 131
- snk\_usb\_comm\_en, 131
- snk\_usb\_susp\_en, 132
- spm\_cfg\_tbl\_offset, 132
- src\_pdo\_cnt, 132
- src\_pdo\_list, 132
- svid\_vdm\_length, 132
- svid\_vdm\_offset, 132
- swap\_response, 132
- tbthost\_cfg\_tbl\_offset, 132
- try\_src\_en, 132
- type\_a\_chg\_tbl\_offset, 133
- type\_a\_enable, 133
- type\_a\_pwr\_tbl\_offset, 133
- uvp\_tbl\_offset, 133
- vconn\_ocp\_tbl\_offset, 133
- vconn\_retain, 133
- pd\_port\_status\_t, 136
  - status, 137
  - val, 137
- pd\_port\_status\_t::PD\_PORT\_STAT, 133
  - ccg\_spec\_rev, 134
  - contract\_exist, 134
  - cur\_data\_role, 134
  - cur\_power\_role, 134
  - dfft\_data\_pref, 134
  - dfft\_data\_role, 134
  - dfft\_power\_pref, 135
  - dfft\_power\_role, 135
  - emca\_present, 135
  - emca\_spec\_rev, 135
  - emca\_type, 135
  - min\_state, 135
  - pe\_rdy, 135
  - peer\_pd3\_supp, 135
  - peer\_unchunk\_supp, 135
  - reserved0, 136
  - reserved2, 136
  - rp\_status, 136
  - vconn\_on, 136
  - vconn\_src, 136
- pd\_power\_status\_t, 137
  - battery\_input, 137
  - dummy, 137
  - event\_flags, 138
  - intl\_temperature, 138
  - power\_status, 138
  - present\_input, 138
  - temp\_status, 138
- pd\_prot\_dis\_bist\_cm2
  - pd\_protocol.h, 416
- pd\_prot\_dis\_bist\_test\_data
  - pd\_protocol.h, 416
- pd\_prot\_en\_bist\_cm2
  - pd\_protocol.h, 416
- pd\_prot\_en\_bist\_test\_data
  - pd\_protocol.h, 417
- pd\_prot\_frs\_rx\_disable
  - pd\_protocol.h, 417
- pd\_prot\_frs\_rx\_enable
  - pd\_protocol.h, 417
- pd\_prot\_frs\_tx\_disable
  - pd\_protocol.h, 418
- pd\_prot\_frs\_tx\_enable
  - pd\_protocol.h, 418
- pd\_prot\_get\_rx\_packet
  - pd\_protocol.h, 418
- pd\_prot\_init
  - pd\_protocol.h, 419
- pd\_prot\_is\_busy
  - pd\_protocol.h, 419
- pd\_prot\_refresh\_roles
  - pd\_protocol.h, 419
- pd\_prot\_reset
  - pd\_protocol.h, 420
- pd\_prot\_reset\_all
  - pd\_protocol.h, 420
- pd\_prot\_reset\_rx
  - pd\_protocol.h, 420
- pd\_prot\_rx\_dis
  - pd\_protocol.h, 421
- pd\_prot\_rx\_en
  - pd\_protocol.h, 421
- pd\_prot\_send\_cable\_reset
  - pd\_protocol.h, 421
- pd\_prot\_send\_ctrl\_msg
  - pd\_protocol.h, 422
- pd\_prot\_send\_data\_msg
  - pd\_protocol.h, 422
- pd\_prot\_send\_extd\_msg
  - pd\_protocol.h, 423
- pd\_prot\_send\_hard\_reset
  - pd\_protocol.h, 423
- pd\_prot\_set\_avoid\_retry
  - pd\_protocol.h, 423
- pd\_prot\_start
  - pd\_protocol.h, 424
- pd\_prot\_stop
  - pd\_protocol.h, 424
- pd\_protocol.h
  - pd\_prot\_dis\_bist\_cm2, 416

- pd\_prot\_dis\_bist\_test\_data, [416](#)
- pd\_prot\_en\_bist\_cm2, [416](#)
- pd\_prot\_en\_bist\_test\_data, [417](#)
- pd\_prot\_frs\_rx\_disable, [417](#)
- pd\_prot\_frs\_rx\_enable, [417](#)
- pd\_prot\_frs\_tx\_disable, [418](#)
- pd\_prot\_frs\_tx\_enable, [418](#)
- pd\_prot\_get\_rx\_packet, [418](#)
- pd\_prot\_init, [419](#)
- pd\_prot\_is\_busy, [419](#)
- pd\_prot\_refresh\_roles, [419](#)
- pd\_prot\_reset, [420](#)
- pd\_prot\_reset\_all, [420](#)
- pd\_prot\_reset\_rx, [420](#)
- pd\_prot\_rx\_dis, [421](#)
- pd\_prot\_rx\_en, [421](#)
- pd\_prot\_send\_cable\_reset, [421](#)
- pd\_prot\_send\_ctrl\_msg, [422](#)
- pd\_prot\_send\_data\_msg, [422](#)
- pd\_prot\_send\_extd\_msg, [423](#)
- pd\_prot\_send\_hard\_reset, [423](#)
- pd\_prot\_set\_avoid\_retry, [423](#)
- pd\_prot\_start, [424](#)
- pd\_prot\_stop, [424](#)
- pd\_remove\_internal\_fb\_res\_div
  - pdss\_hal.h, [502](#)
- pd\_reset\_edge\_det
  - pdss\_hal.h, [502](#)
- pd\_rev\_3
  - pd\_stack\_conf\_t, [139](#)
- pd\_rev\_t
  - pd.h, [395](#)
- pd\_sample\_pfc\_comp
  - pdss\_hal.h, [502](#)
- pd\_set\_pfc\_comp
  - pdss\_hal.h, [503](#)
- pd\_set\_sr\_comp
  - pdss\_hal.h, [503](#)
- pd\_soft\_reset\_reason\_t
  - pd.h, [395](#)
- pd\_srgdrv\_disable
  - pdss\_hal.h, [504](#)
- pd\_srgdrv\_enable
  - pdss\_hal.h, [504](#)
- pd\_sr\_sense\_disable
  - pdss\_hal.h, [504](#)
- pd\_sr\_sense\_enable
  - pdss\_hal.h, [505](#)
- pd\_stack\_conf\_t, [138](#)
  - frs\_rx, [139](#)
  - frs\_tx, [139](#)
  - pd\_rev\_3, [139](#)
  - source\_only, [139](#)
- pd\_status\_t, [139](#)
  - avoid\_retry, [140](#)
  - bist\_test\_en, [140](#)
  - cbk, [140](#)
  - ctrs, [140](#)
  - cur\_rec\_msg\_id, [141](#)
  - frs\_rx\_enable, [141](#)
  - frs\_tx\_enable, [141](#)
  - last\_rcvd\_sop, [141](#)
  - last\_tx\_sop, [141](#)
  - rev3\_enable, [141](#)
  - rx\_busy, [141](#)
  - rx\_evt, [141](#)
  - rx\_packet, [141](#)
  - tx\_buf, [142](#)
  - tx\_busy, [142](#)
  - tx\_count, [142](#)
  - tx\_extd, [142](#)
  - tx\_extd\_hdr, [142](#)
  - tx\_header, [142](#)
  - tx\_msg\_type, [142](#)
  - tx\_sop, [142](#)
- pd\_stop\_pfc\_comp
  - pdss\_hal.h, [505](#)
- pd\_stop\_sr\_comp
  - pdss\_hal.h, [505](#)
- pd\_supply\_change\_cbk\_t
  - pdss\_hal.h, [448](#)
- pd\_support
  - dpm\_status\_t, [83](#)
- pd\_typec\_dis\_dpslp\_rp
  - pdss\_hal.h, [505](#)
- pd\_typec\_dis\_rd
  - pdss\_hal.h, [506](#)
- pd\_typec\_dis\_rp
  - pdss\_hal.h, [506](#)
- pd\_typec\_en\_deadbat\_rd
  - pdss\_hal.h, [506](#)
- pd\_typec\_en\_dpslp\_rp
  - pdss\_hal.h, [507](#)
- pd\_typec\_en\_rd
  - pdss\_hal.h, [507](#)
- pd\_typec\_en\_rp
  - pdss\_hal.h, [507](#)
- pd\_typec\_get\_cc\_status
  - pdss\_hal.h, [508](#)
- pd\_typec\_init
  - pdss\_hal.h, [508](#)
- pd\_typec\_rd\_enable
  - pdss\_hal.h, [508](#)
- pd\_typec\_set\_polarity
  - pdss\_hal.h, [509](#)
- pd\_typec\_snk\_update\_trim
  - pdss\_hal.h, [509](#)
- pd\_typec\_start
  - pdss\_hal.h, [509](#)
- pd\_typec\_stop
  - pdss\_hal.h, [510](#)
- pd\_vconn\_disable
  - pdss\_hal.h, [510](#)
- pd\_vconn\_enable
  - pdss\_hal.h, [510](#)
- pdo.h

- APP\_PPS\_SNK\_CONTRACT\_PERIOD, [302](#)
- APP\_PPS\_SNK\_CONTRACT\_RETRY\_PERIOD, [302](#)
- app\_update\_rdo, [302](#)
- eval\_rdo, [304](#)
- eval\_src\_cap, [304](#)
- pdo\_sel\_alg
  - custom\_host\_cfg\_settings\_t, [67](#)
- pdo\_sel\_alg\_t
  - pd.h, [395](#)
- pdo\_t
  - pd.h, [396](#)
- pdss\_hal.h
  - AUTO\_CTRL\_MESSAGE\_GOODCRC\_MASK\_↔ CFG, [444](#)
  - AUTO\_DATA\_MESSAGE\_GOODCRC\_MASK\_↔ CFG, [444](#)
  - AUTO\_EXTD\_MESSAGE\_GOODCRC\_MASK\_↔ CFG, [444](#)
  - aux\_resistor\_config\_t, [449](#)
  - aux\_resistor\_configure, [455](#)
  - ccg\_bc\_cdp\_en, [456](#)
  - ccg\_bc\_cdp\_sm, [456](#)
  - ccg\_bc\_dcp\_en, [456](#)
  - ccg\_bc\_dis, [457](#)
  - ccg\_bc\_is\_cdp, [457](#)
  - ccg\_config\_dp\_dm\_mux, [457](#)
  - ccg\_is\_cdp\_sm\_busy, [458](#)
  - ccg\_refgen\_op\_t, [449](#)
  - ccg\_set\_fault\_cb, [458](#)
  - ccg\_supply\_t, [450](#)
  - comp\_id\_t, [450](#)
  - comp\_tr\_id\_t, [450](#)
  - dpdm\_mux\_cfg\_t, [451](#)
  - EXPECTED\_GOOD\_CRC\_CLEAR\_MASK, [444](#)
  - EXPECTED\_GOOD\_CRC\_HDR\_DIFF\_MASK\_↔ REV3, [444](#)
  - EXPECTED\_GOOD\_CRC\_HDR\_MASK, [444](#)
  - filter\_edge\_detect\_cfg\_t, [451](#)
  - filter\_id\_t, [451](#)
  - frs\_tx\_source\_t, [452](#)
  - get\_aux1\_resistor\_config, [458](#)
  - get\_aux2\_resistor\_config, [460](#)
  - get\_sbu1\_switch\_state, [460](#)
  - get\_sbu2\_switch\_state, [460](#)
  - hpdt\_ctrl1\_reg\_check\_start, [461](#)
  - lscsa\_app\_config\_t, [452](#)
  - PD\_ADC\_CB\_T, [447](#)
  - PD\_ADC\_ID\_T, [453](#)
  - PD\_ADC\_INPUT\_T, [453](#)
  - PD\_ADC\_INT\_T, [453](#)
  - PD\_ADC\_VREF\_T, [454](#)
  - PD\_MSG\_HDR\_REV2\_IGNORE\_MASK, [444](#)
  - PDSS\_CC\_FILT\_CYCLES, [444](#)
  - PDSS\_DRP\_HIGH\_PERIOD\_MS, [445](#)
  - PDSS\_DRP\_HIGH\_PERIOD\_VAL, [445](#)
  - PDSS\_DRP\_TOGGLE\_PERIOD\_MS, [445](#)
  - PDSS\_DRP\_TOGGLE\_PERIOD\_VAL, [445](#)
  - PDSS\_MAX\_RX\_MEM\_HALF\_SIZE, [445](#)
  - PDSS\_MAX\_RX\_MEM\_SIZE, [445](#)
  - PDSS\_MAX\_TX\_MEM\_HALF\_SIZE, [445](#)
  - PDSS\_MAX\_TX\_MEM\_SIZE, [445](#)
  - PGDO\_PD\_ISINK\_TERMINAL\_VAL\_1, [446](#)
  - PGDO\_PD\_ISINK\_TERMINAL\_VAL, [445](#)
  - pasc\_mode\_t, [452](#)
  - pasc\_ptdrv\_cont\_cbk\_t, [447](#)
  - pasc\_valley\_cbk\_t, [447](#)
  - pd\_adc\_calibrate, [461](#)
  - pd\_adc\_comparator\_ctrl, [461](#)
  - pd\_adc\_comparator\_sample, [462](#)
  - pd\_adc\_free\_run\_ctrl, [462](#)
  - pd\_adc\_get\_comparator\_status, [463](#)
  - pd\_adc\_get\_vbus\_voltage, [463](#)
  - pd\_adc\_init, [464](#)
  - pd\_adc\_level\_to\_volt, [464](#)
  - pd\_adc\_sample, [465](#)
  - pd\_adc\_select\_vref, [465](#)
  - pd\_adc\_volt\_to\_level, [465](#)
  - pd\_cf\_disable, [466](#)
  - pd\_cf\_enable, [466](#)
  - pd\_cf\_get\_status, [467](#)
  - pd\_cf\_mon\_disable, [467](#)
  - pd\_cf\_mon\_enable, [467](#)
  - pd\_cmp\_cbk\_t, [448](#)
  - pd\_cmp\_get\_status, [468](#)
  - pd\_connect\_vbus\_div\_to\_amux, [468](#)
  - pd\_disconnect\_ra, [468](#)
  - pd\_disconnect\_vbus\_div\_from\_amux, [469](#)
  - pd\_enable\_vconn\_comp, [469](#)
  - pd\_fet\_automode\_disable, [469](#)
  - pd\_fet\_automode\_enable, [470](#)
  - pd\_fet\_dr\_t, [454](#)
  - pd\_frs\_rx\_disable, [470](#)
  - pd\_frs\_rx\_enable, [470](#)
  - pd\_frs\_tx\_disable, [471](#)
  - pd\_frs\_tx\_enable, [471](#)
  - pd\_get\_vbus\_adc\_level, [471](#)
  - pd\_get\_vconn\_status, [472](#)
  - pd\_hal\_abort\_auto\_toggle, [472](#)
  - pd\_hal\_cleanup, [472](#)
  - pd\_hal\_config\_auto\_toggle, [473](#)
  - pd\_hal\_dual\_fet\_config, [473](#)
  - pd\_hal\_enable\_internal\_vbus\_mon, [474](#)
  - pd\_hal\_get\_vbus\_csa\_rsense, [474](#)
  - pd\_hal\_get\_vbus\_detach\_adc, [474](#)
  - pd\_hal\_get\_vbus\_detach\_input, [474](#)
  - pd\_hal\_init, [475](#)
  - pd\_hal\_is\_auto\_toggle\_active, [475](#)
  - pd\_hal\_is\_sink\_fet\_on, [475](#)
  - pd\_hal\_measure\_ea\_volt, [475](#)
  - pd\_hal\_measure\_line\_volt, [476](#)
  - pd\_hal\_measure\_vbus, [476](#)
  - pd\_hal\_measure\_vbus\_cur, [476](#)
  - pd\_hal\_measure\_vbus\_in, [477](#)
  - pd\_hal\_set\_cc\_ovp\_pending, [477](#)
  - pd\_hal\_set\_fet\_drive, [477](#)

pd\_hal\_set\_reference, 478  
 pd\_hal\_set\_supply\_change\_evt\_cb, 478  
 pd\_hal\_set\_vbus\_csa\_rsense, 479  
 pd\_hal\_set\_vbus\_detach\_params, 479  
 pd\_hal\_set\_vbus\_mon\_divider, 479  
 pd\_hal\_typec\_sm\_restart, 480  
 pd\_hal\_vconn\_ocp\_disable, 480  
 pd\_hal\_vconn\_ocp\_enable, 480  
 pd\_internal\_cfet\_off, 481  
 pd\_internal\_cfet\_on, 481  
 pd\_internal\_cfet\_soft\_start\_off, 482  
 pd\_internal\_cfet\_soft\_start\_on, 482  
 pd\_internal\_pfet\_off, 482  
 pd\_internal\_pfet\_on, 483  
 pd\_internal\_pfet\_soft\_start\_off, 483  
 pd\_internal\_pfet\_soft\_start\_on, 483  
 pd\_internal\_vbus\_discharge\_off, 484  
 pd\_internal\_vbus\_discharge\_on, 484  
 pd\_internal\_vbus\_in\_discharge\_off, 484  
 pd\_internal\_vbus\_in\_discharge\_on, 485  
 pd\_internal\_vbus\_load\_change\_isr\_enable, 485  
 pd\_internal\_vbus\_ocp\_dis, 486  
 pd\_internal\_vbus\_ocp\_en, 486  
 pd\_internal\_vbus\_ovp\_dis, 486  
 pd\_internal\_vbus\_ovp\_en, 487  
 pd\_internal\_vbus\_rcp\_dis, 487  
 pd\_internal\_vbus\_rcp\_en, 488  
 pd\_internal\_vbus\_scp\_dis, 488  
 pd\_internal\_vbus\_scp\_en, 489  
 pd\_internal\_vbus\_uvp\_dis, 489  
 pd\_internal\_vbus\_uvp\_en, 489  
 pd\_is\_v5v\_supply\_on, 490  
 pd\_is\_vconn\_present, 490  
 pd\_lscsa\_calc\_cfg, 491  
 pd\_lscsa\_cfg, 491  
 pd\_pasc\_get\_valley, 491  
 pd\_pasc\_lp\_disable, 492  
 pd\_pasc\_lp\_ds\_allowed, 492  
 pd\_pasc\_lp\_enable, 492  
 pd\_pasc\_lp\_is\_active, 493  
 pd\_pasc\_lp\_task, 493  
 pd\_pasc\_poll\_task, 493  
 pd\_pasc\_ptdrv\_cont\_det\_disable, 494  
 pd\_pasc\_ptdrv\_cont\_det\_enable, 494  
 pd\_pasc\_send\_stop\_signal, 494  
 pd\_pasc\_set\_valley, 495  
 pd\_pasc\_start, 495  
 pd\_pasc\_valley\_algo\_enable, 495  
 pd\_pasc\_vbus\_in\_set\_volt, 496  
 pd\_pasc\_vbus\_in\_set\_volt\_abort, 496  
 pd\_pasc\_vbus\_in\_set\_vsafe\_5V, 496  
 pd\_phy\_abort\_bist\_cm2, 497  
 pd\_phy\_abort\_tx\_msg, 497  
 pd\_phy\_cbk\_t, 448  
 pd\_phy\_deepsleep, 497  
 pd\_phy\_dis\_unchunked\_tx, 498  
 pd\_phy\_en\_unchunked\_tx, 498  
 pd\_phy\_evt\_t, 454  
 pd\_phy\_get\_rx\_packet, 498  
 pd\_phy\_init, 499  
 pd\_phy\_is\_busy, 499  
 pd\_phy\_load\_msg, 499  
 pd\_phy\_refresh\_roles, 500  
 pd\_phy\_reset\_rx\_tx\_sm, 500  
 pd\_phy\_send\_bist\_cm2, 501  
 pd\_phy\_send\_msg, 501  
 pd\_phy\_send\_reset, 501  
 pd\_phy\_wakeup, 502  
 pd\_remove\_internal\_fb\_res\_div, 502  
 pd\_reset\_edge\_det, 502  
 pd\_sample\_pfc\_comp, 502  
 pd\_set\_pfc\_comp, 503  
 pd\_set\_sr\_comp, 503  
 pd\_srgdrv\_disable, 504  
 pd\_srgdrv\_enable, 504  
 pd\_srsense\_disable, 504  
 pd\_srsense\_enable, 505  
 pd\_stop\_pfc\_comp, 505  
 pd\_stop\_sr\_comp, 505  
 pd\_supply\_change\_cbk\_t, 448  
 pd\_typec\_dis\_dpslp\_rp, 505  
 pd\_typec\_dis\_rd, 506  
 pd\_typec\_dis\_rp, 506  
 pd\_typec\_en\_deadbat\_rd, 506  
 pd\_typec\_en\_dpslp\_rp, 507  
 pd\_typec\_en\_rd, 507  
 pd\_typec\_en\_rp, 507  
 pd\_typec\_get\_cc\_status, 508  
 pd\_typec\_init, 508  
 pd\_typec\_rd\_enable, 508  
 pd\_typec\_set\_polarity, 509  
 pd\_typec\_snk\_update\_trim, 509  
 pd\_typec\_start, 509  
 pd\_typec\_stop, 510  
 pd\_vconn\_disable, 510  
 pd\_vconn\_enable, 510  
 RX\_CNT\_MAX\_VAL, 446  
 RX\_UI\_BOUNDARY\_DELTA\_VAL, 446  
 sbu\_switch\_configure, 511  
 sbu\_switch\_state\_t, 454  
 VBUS\_OCP\_GPIO\_ACTIVE\_HIGH, 446  
 VBUS\_OCP\_GPIO\_ACTIVE\_LOW, 446  
 VBUS\_OCP\_MODE\_EXT, 446  
 VBUS\_OCP\_MODE\_INT\_AUTOCTRL, 446  
 VBUS\_OCP\_MODE\_INT\_SW\_DB, 446  
 VBUS\_OCP\_MODE\_INT, 446  
 VBUS\_OCP\_MODE\_POLLING, 447  
 vbus\_cf\_cbk\_t, 448  
 vbus\_load\_chg\_cbk\_t, 449  
 vbus\_ovp\_mode\_t, 455  
 vbus\_uvp\_mode\_t, 455  
 pe\_cbl\_state\_t  
   pd.h, 396  
 pe\_clear\_hard\_reset\_count  
   pd\_policy\_engine.h, 412  
 pe\_disabled

- pd\_policy\_engine.h, 413
- pe\_evt
  - dpm\_status\_t, 83
- pe\_fsm
  - pd\_policy\_engine.h, 413
- pe\_fsm\_state
  - dpm\_status\_t, 83
- pe\_fsm\_state\_t
  - pd.h, 396
- pe\_get\_pps\_status
  - pd\_policy\_engine.h, 413
- pe\_init
  - pd\_policy\_engine.h, 414
- pe\_is\_busy
  - pd\_policy\_engine.h, 414
- pe\_push\_to\_buf
  - pd\_policy\_engine.h, 414
- pe\_rdy
  - pd\_port\_status\_t::PD\_PORT\_STAT, 135
- pe\_start
  - pd\_policy\_engine.h, 414
- pe\_stop
  - pd\_policy\_engine.h, 415
- peak\_cur\_cap\_t
  - pd.h, 397
- peer\_pd3\_supp
  - pd\_port\_status\_t::PD\_PORT\_STAT, 135
- peer\_unchunk\_supp
  - pd\_port\_status\_t::PD\_PORT\_STAT, 135
- period
  - ccg\_timer\_t, 62
- phy\_conn
  - pd\_do\_t::ACT\_CBL\_VDO\_2, 26
- pk\_current
  - pd\_do\_t::FIXED\_SRC, 95
- polarity
  - dpm\_status\_t, 83
- policy\_mgr\_en
  - auto\_cfg\_settings\_t, 50
- port\_cap
  - pd\_do\_t::STD\_DP\_VDO, 173
- port\_conf
  - pd\_config\_t, 110
- port\_disable
  - dpm\_status\_t, 83
  - pd\_port\_config\_t, 129
- port\_numb
  - pd\_do\_t::DFP\_VDO, 68
- port\_pwr
  - auto\_cfg\_settings\_t, 50
- port\_role
  - dpm\_status\_t, 83
  - pd\_port\_config\_t, 129
- port\_role\_t
  - pd.h, 398
- port\_status
  - dpm\_status\_t, 83
- port\_type\_t
  - pd.h, 398
- power\_status
  - pd\_power\_status\_t, 138
- pps\_en
  - auto\_cfg\_settings\_t, 50
- pps\_pwr\_limited
  - pd\_do\_t::PPS\_SRC, 145
- pps\_snk
  - pd\_do\_t, 115
- pps\_src
  - pd\_do\_t, 115
- pps\_src\_en
  - dpm\_status\_t, 83
- pps\_status
  - dpm\_status\_t, 84
- pref\_data\_role
  - tbthost\_cfg\_settings\_t, 185
- pref\_pwr\_role
  - tbthost\_cfg\_settings\_t, 185
- present\_input
  - pd\_power\_status\_t, 138
- prim\_sec\_turns\_ratio
  - pwr\_params\_t, 148
- prl\_cntrs\_t, 146
  - first\_msg\_rcvd, 146
  - rec\_msg\_id, 146
  - tr\_msg\_id, 146
- prod\_type
  - pd\_do\_t::STD\_VDM\_ID\_HDR, 177
- prod\_type\_dfp
  - pd\_do\_t::STD\_VDM\_ID\_HDR, 177
- protection\_enable
  - pd\_port\_config\_t, 129
- psink.h
  - psnk\_disable, 305
  - psnk\_enable, 306
  - psnk\_set\_current, 306
  - psnk\_set\_voltage, 306
- psnk\_cur
  - app\_status\_t, 46
- psnk\_disable
  - app\_cbk\_t, 40
  - psink.h, 305
- psnk\_enable
  - app\_cbk\_t, 40
  - psink.h, 306
- psnk\_set\_current
  - app\_cbk\_t, 40
  - psink.h, 306
- psnk\_set\_voltage
  - app\_cbk\_t, 40
  - psink.h, 306
- psnk\_volt
  - app\_status\_t, 46
- psource.h
  - PPS\_CF\_VBUS\_DECREMENT\_STEP, 307
  - psrc\_disable, 307
  - psrc\_enable, 308

- psrc\_get\_voltage, 308
- psrc\_set\_current, 309
- psrc\_set\_voltage, 309
- psrc\_disable
  - app\_cbk\_t, 40
  - psource.h, 307
- psrc\_enable
  - app\_cbk\_t, 40
  - psource.h, 308
- psrc\_get\_voltage
  - app\_cbk\_t, 40
  - psource.h, 308
- psrc\_rising
  - app\_status\_t, 46
- psrc\_set\_current
  - app\_cbk\_t, 40
  - psource.h, 309
- psrc\_set\_voltage
  - app\_cbk\_t, 41
  - psource.h, 309
- psrc\_volt
  - app\_status\_t, 46
- psrc\_volt\_old
  - app\_status\_t, 47
- pwm\_fix\_freq
  - pwr\_params\_t, 148
- pwm\_max\_freq
  - pwr\_params\_t, 148
- pwm\_min\_freq
  - pwr\_params\_t, 148
- pwm\_mode
  - pwr\_params\_t, 148
- pwr\_limited
  - dpm\_status\_t, 84
- pwr\_low
  - pd\_do\_t::DP\_STATUS\_VDO, 71
- pwr\_params\_t, 146
  - cable\_resistance, 147
  - cur\_sense\_res, 147
  - fb\_ctrl\_r1, 147
  - fb\_ctrl\_r2, 147
  - fb\_type, 148
  - max\_pwm\_duty\_cycle, 148
  - prim\_sec\_turns\_ratio, 148
  - pwm\_fix\_freq, 148
  - pwm\_max\_freq, 148
  - pwm\_min\_freq, 148
  - pwm\_mode, 148
  - reserved\_0, 148
  - reserved\_2, 148
  - reserved\_3, 149
  - sr\_async\_thresh, 149
  - sr\_enable, 149
  - sr\_fall\_time, 149
  - sr\_rise\_time, 149
  - sr\_supply\_doubler, 149
  - src\_gate\_drv\_str, 149
  - table\_len, 149
  - vbtr\_down\_step\_width, 149
  - vbtr\_up\_step\_width, 150
  - vbus\_dflt\_volt, 150
  - vbus\_max\_volt, 150
  - vbus\_min\_volt, 150
  - vbus\_offset\_volt, 150
- pwr\_ready\_cbk
  - app\_status\_t, 47
- pwr\_ready\_cbk\_t
  - pd.h, 383
- pwr\_role
  - pd\_hdr\_t::PD\_HDR, 120
- pwr\_tbl\_offset
  - pd\_port\_config\_t, 130
- pwr\_threshold
  - custom\_host\_cfg\_settings\_t, 67
- QC3\_MIN\_VOLT
  - battery\_charging.h, 265
- QC\_AMP\_12V
  - battery\_charging.h, 266
- QC\_AMP\_20V
  - battery\_charging.h, 266
- QC\_AMP\_5V
  - battery\_charging.h, 266
- QC\_AMP\_9V
  - battery\_charging.h, 266
- QC\_AMP\_CONT
  - battery\_charging.h, 266
- QC\_CONT\_VOLT\_CHANGE\_PER\_PULSE
  - battery\_charging.h, 266
- qc\_4\_0\_data\_vdo
  - pd\_do\_t, 115
- qc\_set\_cf\_limit
  - battery\_charging.h, 275
- qc\_src\_type
  - chg\_cfg\_params\_t, 63
- RDO\_IDX
  - pd.h, 377
- REG\_FIELD\_GET
  - utils.h, 569
- REG\_FIELD\_UPDATE
  - utils.h, 570
- REMOVE\_FLAG
  - alt\_modes\_mgr.h, 206
- RETIMER\_CFG\_AVAILABLE
  - bb\_retimer.h, 278
- RETIMER\_CFG\_SLAVE\_ADDR
  - bb\_retimer.h, 278
- RETIMER\_I2C\_TIMEOUT
  - bb\_retimer.h, 278
- REV\_BYTE\_ORDER
  - utils.h, 570
- RIDGE\_CMD\_CCG\_RESET
  - ridge\_slave.h, 297
- RIDGE\_CMD\_INT\_CLEAR
  - ridge\_slave.h, 297
- RIDGE\_INIT\_HPD\_DEQUEUE\_TIMER\_PERIOD



- app.h, [241](#)
- RIDGE\_IRQ\_ACK
  - ridge\_slave.h, [297](#)
- RIDGE\_SLAVE\_MAX\_READ\_SIZE
  - ridge\_slave.h, [298](#)
- RIDGE\_SLAVE\_MAX\_WRITE\_SIZE
  - ridge\_slave.h, [298](#)
- RIDGE\_SLAVE\_MIN\_WRITE\_SIZE
  - ridge\_slave.h, [298](#)
- RIDGE\_SLAVE\_SCB\_CLOCK\_FREQ
  - ridge\_slave.h, [298](#)
- RIDGE\_SLAVE\_SCB\_INDEX
  - ridge\_slave.h, [298](#)
- RX\_CNT\_MAX\_VAL
  - pdss\_hal.h, [446](#)
- RX\_UI\_BOUNDARY\_DELTA\_VAL
  - pdss\_hal.h, [446](#)
- ra\_present
  - dpm\_status\_t, [84](#)
- ra\_timeout
  - pd\_port\_config\_t, [130](#)
- rand\_base
  - dpm\_status\_t, [84](#)
- rcp\_settings\_t, [151](#)
  - resvd, [151](#)
  - retry\_cnt, [152](#)
  - table\_len, [152](#)
- rcp\_tbl\_offset
  - pd\_port\_config\_t, [130](#)
- rd\_cc\_status\_t
  - pd.h, [398](#)
- rdo
  - pd\_contract\_info\_t, [111](#)
- rdo\_bat
  - pd\_do\_t, [115](#)
- rdo\_bat\_gvb
  - pd\_do\_t, [115](#)
- rdo\_fix\_var
  - pd\_do\_t, [115](#)
- rdo\_fix\_var\_gvb
  - pd\_do\_t, [115](#)
- rdo\_gen
  - pd\_do\_t, [115](#)
- rdo\_gen\_gvb
  - pd\_do\_t, [116](#)
- rdo\_pps
  - pd\_do\_t, [116](#)
- rdo\_type\_t
  - pd.h, [398](#)
- rec\_msg\_id
  - prl\_cntrs\_t, [146](#)
- recep
  - pd\_do\_t::STD\_DP\_VDO, [173](#)
- reg\_alt\_mode\_mgr
  - alt\_modes\_mgr.h, [215](#)
- reg\_am\_ptr
  - reg\_am\_t, [164](#)
- reg\_am\_t, [164](#)
  - reg\_am\_ptr, [164](#)
  - svid, [164](#)
- reg\_dp\_modes
  - dp\_sid.h, [225](#)
- reg\_hpi\_modes
  - custom\_hpi\_vid.h, [217](#)
- reg\_intel\_modes
  - intel\_vid.h, [296](#)
- req\_max\_pwr
  - custom\_host\_cfg\_settings\_t, [67](#)
- req\_status
  - app\_resp\_t, [42](#)
- request
  - pd\_extd\_hdr\_t::EXTD\_HDR\_T, [92](#)
  - pd\_hdr\_t::PD\_HDR, [120](#)
- reserved
  - custom\_host\_cfg\_settings\_t, [67](#)
  - fw\_img\_status\_t::fw\_mode\_reason\_t, [97](#)
  - pd\_do\_t::FIXED\_SRC, [95](#)
- reserved0
  - custom\_alt\_cfg\_settings\_t, [66](#)
  - icl\_tgl\_cfg\_settings\_t, [100](#)
  - pd\_port\_status\_t::PD\_PORT\_STAT, [136](#)
- reserved1
  - custom\_alt\_cfg\_settings\_t, [66](#)
  - fw\_img\_status\_t::fw\_mode\_reason\_t, [97](#)
  - sys\_fw\_metadata\_t, [180](#)
- reserved2
  - pd\_port\_status\_t::PD\_PORT\_STAT, [136](#)
  - sys\_fw\_metadata\_t, [180](#)
- reserved\_0
  - auto\_cfg\_settings\_t, [51](#)
  - bat\_chg\_params\_t, [52](#)
  - chg\_cfg\_params\_t, [64](#)
  - pasc\_valley\_table\_t, [107](#)
  - pd\_config\_t, [110](#)
  - pd\_port\_config\_t, [130](#)
  - pwr\_params\_t, [148](#)
- reserved\_1
  - bat\_chg\_params\_t, [52](#)
  - bb\_settings\_t, [57](#)
  - chg\_cfg\_params\_t, [64](#)
  - otp\_settings\_t, [103](#)
  - pd\_port\_config\_t, [130](#)
- reserved\_10
  - pd\_port\_config\_t, [130](#)
- reserved\_11
  - pd\_port\_config\_t, [130](#)
- reserved\_2
  - pd\_config\_t, [110](#)
  - pwr\_params\_t, [148](#)
- reserved\_3
  - dpm\_status\_t, [84](#)
  - pd\_port\_config\_t, [130](#)
  - pwr\_params\_t, [149](#)
- reserved\_4
  - pd\_port\_config\_t, [130](#)
- reserved\_6

- pd\_port\_config\_t, 131
- reserved\_8
  - pd\_port\_config\_t, 131
- reserved\_9
  - pd\_port\_config\_t, 131
- reset\_alt\_mode\_info
  - alt\_modes\_mngr.h, 215
- resp\_buf
  - vdm\_resp\_t, 194
- resp\_do
  - app\_resp\_t, 42
- resp\_status\_t
  - pd.h, 399
- restart\_val
  - otp\_settings\_t, 103
- restart\_val\_1
  - otp\_settings\_t, 103
- resvd
  - rcp\_settings\_t, 151
- retimer\_clr\_evt
  - bb\_retimer.h, 279
- retimer\_dis\_req
  - app\_status\_t, 47
- retimer\_disable
  - bb\_retimer.h, 279
- retimer\_enable
  - bb\_retimer.h, 280
- retimer\_force\_enable
  - bb\_retimer.h, 280
- retimer\_init
  - bb\_retimer.h, 280
- retimer\_is\_present
  - bb\_retimer.h, 281
- retimer\_read\_wrapper
  - bb\_retimer.h, 281
- retimer\_reg\_write
  - bb\_retimer.h, 281
- retimer\_set\_compl\_mode
  - bb\_retimer.h, 282
- retimer\_set\_evt
  - bb\_retimer.h, 282
- retimer\_set\_slave\_address
  - bb\_retimer.h, 282
- retimer\_sleep\_allowed
  - bb\_retimer.h, 283
- retimer\_start\_debug\_poll
  - bb\_retimer.h, 283
- retimer\_status\_update
  - bb\_retimer.h, 283
- retimer\_task
  - bb\_retimer.h, 284
- retimer\_update\_is\_pending
  - bb\_retimer.h, 284
- retimer\_write\_wrapper
  - bb\_retimer.h, 284
- retry\_cnt
  - ocp\_settings\_t, 101
  - ovp\_settings\_t, 104
- rcp\_settings\_t, 152
- scp\_settings\_t, 165
- uvp\_settings\_t, 190
- vconn\_ocp\_settings\_t, 192
- rev3\_en
  - dpm\_status\_t, 84
- rev3\_enable
  - pd\_status\_t, 141
- rev\_pol
  - dpm\_status\_t, 84
- ridge\_eval\_cmd
  - intel\_ridge.h, 290
- ridge\_force\_status\_update
  - intel\_ridge.h, 290
- ridge\_reg\_reset
  - ridge\_slave.h, 299
- ridge\_set\_mux
  - intel\_ridge.h, 290
- ridge\_set\_vpro
  - intel\_ridge.h, 291
- ridge\_slave.h
  - ICL\_SOC\_ACK\_TIMEOUT\_PERIOD, 297
  - PMC\_SLAVE\_ADDR\_PORT0, 297
  - PMC\_SLAVE\_ADDR\_PORT1, 297
  - RIDGE\_CMD\_CCG\_RESET, 297
  - RIDGE\_CMD\_INT\_CLEAR, 297
  - RIDGE\_IRQ\_ACK, 297
  - RIDGE\_SLAVE\_MAX\_READ\_SIZE, 298
  - RIDGE\_SLAVE\_MAX\_WRITE\_SIZE, 298
  - RIDGE\_SLAVE\_MIN\_WRITE\_SIZE, 298
  - RIDGE\_SLAVE\_SCB\_CLOCK\_FREQ, 298
  - RIDGE\_SLAVE\_SCB\_INDEX, 298
  - ridge\_reg\_reset, 299
  - ridge\_slave\_deinit, 299
  - ridge\_slave\_init, 299
  - ridge\_slave\_is\_host\_connected, 299
  - ridge\_slave\_reg\_addr\_t, 298
  - ridge\_slave\_set\_ocp\_status, 300
  - ridge\_slave\_sleep, 300
  - ridge\_slave\_soc\_timeout\_event\_control, 300
  - ridge\_slave\_status\_update, 301
  - ridge\_slave\_task, 301
  - ridge\_update\_is\_pending, 301
- ridge\_slave\_deinit
  - ridge\_slave.h, 299
- ridge\_slave\_init
  - ridge\_slave.h, 299
- ridge\_slave\_is\_host\_connected
  - ridge\_slave.h, 299
- ridge\_slave\_reg\_addr\_t
  - ridge\_slave.h, 298
- ridge\_slave\_set\_ocp\_status
  - ridge\_slave.h, 300
- ridge\_slave\_sleep
  - ridge\_slave.h, 300
- ridge\_slave\_soc\_timeout\_event\_control
  - ridge\_slave.h, 300
- ridge\_slave\_status\_update

- ridge\_slave.h, 301
- ridge\_slave\_task
  - ridge\_slave.h, 301
- ridge\_update\_dr
  - intel\_ridge.h, 291
- ridge\_update\_is\_pending
  - ridge\_slave.h, 301
- role\_at\_connect
  - dpm\_status\_t, 84
- rp\_cc\_status\_t
  - pd.h, 399
- rp\_detach\_disable
  - custom\_host\_cfg\_settings\_t, 67
- rp\_status
  - pd\_port\_status\_t::PD\_PORT\_STAT, 136
- rp\_supported
  - dpm\_status\_t, 84
  - pd\_port\_config\_t, 131
- rp\_term\_t
  - pd.h, 399
- rsrsvd
  - pd\_do\_t::FIXED\_SNK, 93
- rsrsvd1
  - pd\_do\_t::RDO\_BAT\_GIVEBACK, 155
  - pd\_do\_t::RDO\_BAT, 153
  - pd\_do\_t::RDO\_FIXED\_VAR\_GIVEBACK, 158
  - pd\_do\_t::RDO\_FIXED\_VAR, 156
  - pd\_do\_t::RDO\_GEN\_GVB, 161
  - pd\_do\_t::RDO\_GEN, 160
- rsrsvd2
  - pd\_do\_t::RDO\_BAT\_GIVEBACK, 155
  - pd\_do\_t::RDO\_BAT, 153
  - pd\_do\_t::RDO\_FIXED\_VAR\_GIVEBACK, 158
  - pd\_do\_t::RDO\_FIXED\_VAR, 156
  - pd\_do\_t::RDO\_GEN\_GVB, 161
  - pd\_do\_t::RDO\_GEN, 160
- rsvd
  - pd\_do\_t::DP\_STATUS\_VDO, 71
  - pd\_do\_t::STD\_DP\_VDO, 173
- rsvd0
  - pd\_do\_t::ACT\_CBL\_VDO\_2, 26
  - pd\_do\_t::DFP\_VDO, 68
  - pd\_do\_t::ENTERUSB\_VDO, 90
  - pd\_do\_t::TBT\_UFP\_VDO, 182
  - pd\_do\_t::UFP\_VDO\_1, 187
  - tbthost\_cfg\_settings\_t, 185
- rsvd1
  - pd\_do\_t::ACT\_CBL\_VDO\_1, 24
  - pd\_do\_t::ACT\_CBL\_VDO\_2, 26
  - pd\_do\_t::ACT\_CBL\_VDO, 22
  - pd\_do\_t::ADO\_ALERT, 29
  - pd\_do\_t::BIST\_DO, 61
  - pd\_do\_t::DFP\_VDO, 68
  - pd\_do\_t::DP\_CONFIG\_VDO, 69
  - pd\_do\_t::ENTERUSB\_VDO, 91
  - pd\_do\_t::PAS\_CBL\_VDO, 106
  - pd\_do\_t::PPS\_SNK, 143
  - pd\_do\_t::PPS\_SRC, 145
- pd\_do\_t::RDO\_PPS, 163
- pd\_do\_t::STD\_AMA\_VDO\_PD3, 169
- pd\_do\_t::STD\_AMA\_VDO, 167
- pd\_do\_t::STD\_CBL\_VDO, 171
- pd\_do\_t::STD\_VDM\_HDR, 176
- pd\_do\_t::STD\_VDM\_ID\_HDR, 177
- pd\_do\_t::TBT\_CBL\_VDO, 181
- pd\_do\_t::TBT\_UFP\_VDO, 182
- pd\_do\_t::TBT\_VDO, 184
- pd\_do\_t::UFP\_VDO\_1, 187
- pd\_do\_t::USTD\_VDM\_HDR, 189
- pd\_extd\_hdr\_t::EXTD\_HDR\_T, 92
- pd\_hdr\_t::PD\_HDR, 120
- rsvd2
  - pd\_do\_t::ACT\_CBL\_VDO\_1, 24
  - pd\_do\_t::ACT\_CBL\_VDO, 22
  - pd\_do\_t::ADO\_ALERT, 29
  - pd\_do\_t::BIST\_DO, 61
  - pd\_do\_t::DP\_CONFIG\_VDO, 69
  - pd\_do\_t::ENTERUSB\_VDO, 91
  - pd\_do\_t::PAS\_CBL\_VDO, 106
  - pd\_do\_t::PPS\_SNK, 144
  - pd\_do\_t::PPS\_SRC, 145
  - pd\_do\_t::RDO\_PPS, 163
  - pd\_do\_t::STD\_VDM\_HDR, 176
  - pd\_do\_t::TBT\_VDO, 184
- rsvd3
  - pd\_do\_t::ENTERUSB\_VDO, 91
  - pd\_do\_t::PAS\_CBL\_VDO, 106
  - pd\_do\_t::PPS\_SNK, 144
  - pd\_do\_t::PPS\_SRC, 145
  - pd\_do\_t::RDO\_PPS, 163
- rsvd4
  - pd\_do\_t::PPS\_SNK, 144
  - pd\_do\_t::RDO\_PPS, 163
- rt\_evt\_t
  - bb\_retimer.h, 279
- rx\_busy
  - pd\_status\_t, 141
- rx\_evt
  - pd\_status\_t, 141
- rx\_packet
  - pd\_status\_t, 141
- SET\_FLAG
  - alt\_modes\_mngr.h, 206
- SNK\_DETACH\_VBUS\_POLL\_COUNT
  - pd.h, 378
- SNK\_MIN\_MAX\_MASK
  - pd.h, 378
- SRC\_CUR\_LEVEL\_1\_5A
  - ucsi.h, 579
- SRC\_CUR\_LEVEL\_3A
  - ucsi.h, 579
- SRC\_CUR\_LEVEL\_DEF
  - ucsi.h, 579
- SRC\_DRP\_MIN\_DC
  - pd.h, 378
- SROM\_API\_PARAM\_SIZE

- flash.h, 527
- STATUS\_UPDATE\_VDO
  - dp\_sid.h, 223
- STD\_SVID
  - pd.h, 378
  - vdm\_task\_mngr.h, 227
- STD\_VDM\_VERSION\_IDX
  - pd.h, 378
- STD\_VDM\_VERSION\_REV2
  - pd.h, 378
- STD\_VDM\_VERSION\_REV3
  - pd.h, 378
- STD\_VDM\_VERSION
  - pd.h, 378
- SYS\_APP\_VERSION\_OFFSET
  - system.h, 548
- SYS\_BOOT\_MODE\_RQT\_SIG
  - system.h, 548
- SYS\_BOOT\_TYPE\_APP\_PRIORITY\_POS
  - system.h, 548
- SYS\_BOOT\_TYPE\_FIELD\_OFFSET
  - system.h, 548
- SYS\_BOOT\_TYPE\_FW\_UPDATE\_INTERFACE\_MASK
  - system.h, 549
- SYS\_BOOT\_TYPE\_FW\_UPDATE\_INTERFACE\_POS
  - system.h, 549
- SYS\_BOOT\_TYPE\_SECURE\_BOOT\_MASK
  - system.h, 549
- SYS\_BOOT\_VERSION\_ADDRESS
  - system.h, 549
- SYS\_CONFIG\_TABLE\_SIGN
  - system.h, 549
- SYS\_FW\_CUSTOM\_INFO\_OFFSET
  - system.h, 549
- SYS\_FW\_VERSION\_OFFSET
  - system.h, 549
- SYS\_INVALID\_FW\_START\_ADDR
  - system.h, 549
- SYS\_METADATA\_VALID\_SIG
  - system.h, 549
- SYS\_PSEUDO\_METADATA\_VALID\_SIG
  - system.h, 550
- SYS\_SILICON\_ID\_MASK
  - system.h, 550
- SYS\_SILICON\_ID\_OFFSET
  - system.h, 550
- safe\_valley
  - pasc\_valley\_table\_t, 107
- sbu\_config
  - tbthost\_cfg\_settings\_t, 185
- sbu\_supp
  - pd\_do\_t::ACT\_CBL\_VDO\_1, 24
- sbu\_switch\_configure
  - pdss\_hal.h, 511
- sbu\_switch\_state\_t
  - pdss\_hal.h, 454
- sbu\_type
  - pd\_do\_t::ACT\_CBL\_VDO\_1, 24
- scb/i2c.h, 511
- scp\_settings\_t, 164
  - debounce, 164
  - retry\_cnt, 165
  - table\_len, 165
  - threshold, 165
- scp\_tbl\_offset
  - pd\_port\_config\_t, 131
- sel\_conf
  - pd\_do\_t::DP\_CONFIG\_VDO, 69
- sense\_res
  - ocp\_settings\_t, 101
- sensor\_ctrl
  - sensor\_data\_t, 165
- sensor\_data
  - auto\_cfg\_settings\_t, 51
- sensor\_data\_t, 165
  - sensor\_ctrl, 165
  - sensor\_oc1, 165
  - sensor\_oc2, 166
  - sensor\_oc3, 166
- sensor\_oc1
  - sensor\_data\_t, 165
- sensor\_oc2
  - sensor\_data\_t, 166
- sensor\_oc3
  - sensor\_data\_t, 166
- seq\_num
  - pd\_do\_t::USTD\_VDM\_HDR, 189
- set\_alt\_mode\_mask
  - alt\_modes\_mngr.h, 215
- set\_custom\_svid
  - alt\_modes\_mngr.h, 216
- set\_mux
  - alt\_mode\_hw.h, 201
- set\_mux\_isolate
  - alt\_mode\_info\_t, 36
- set\_retimer\_status
  - bb\_retimer.h, 285
- sflash\_row\_read
  - flash.h, 532
- sflash\_row\_write
  - flash.h, 532
- shutdown\_temp
  - pd\_do\_t::ACT\_CBL\_VDO\_2, 27
- sign
  - pd\_do\_t::DP\_CONFIG\_VDO, 69
- signal
  - pd\_do\_t::STD\_DP\_VDO, 174
- sink\_discharge\_off\_cbk\_t
  - pd.h, 383
- skip\_mux\_config
  - app\_status\_t, 47
- skip\_scan
  - dpm\_status\_t, 85
- slave\_address
  - i2c\_scb\_config\_t, 99

- slave\_mask
  - i2c\_scb\_config\_t, 99
- sln\_pd\_event\_handler
  - app.h, 257
- snk\_cur\_level
  - dpm\_status\_t, 85
- snk\_dis\_cbk
  - app\_status\_t, 47
- snk\_max\_min
  - dpm\_status\_t, 85
- snk\_path\_enable
  - custom\_host\_cfg\_settings\_t, 67
- snk\_pdo
  - dpm\_status\_t, 85
- snk\_pdo\_cnt
  - pd\_port\_config\_t, 131
- snk\_pdo\_count
  - dpm\_status\_t, 85
- snk\_pdo\_list
  - pd\_port\_config\_t, 131
- snk\_pdo\_mask
  - dpm\_status\_t, 85
- snk\_pdo\_max\_min\_current\_pwr
  - pd\_port\_config\_t, 131
- snk\_period
  - dpm\_status\_t, 85
- snk\_rdo
  - dpm\_status\_t, 85
- snk\_rp\_detach\_en
  - dpm\_status\_t, 86
- snk\_sel
  - chg\_cfg\_params\_t, 64
- snk\_sel\_pdo
  - dpm\_status\_t, 86
- snk\_usb\_comm\_en
  - dpm\_status\_t, 86
  - pd\_port\_config\_t, 131
- snk\_usb\_susp\_en
  - dpm\_status\_t, 86
  - pd\_port\_config\_t, 132
- soc\_mux\_config\_delay
  - icl\_tgl\_cfg\_settings\_t, 100
- soc\_mux\_init\_delay
  - icl\_tgl\_cfg\_settings\_t, 100
- sop
  - pd\_packet\_extd\_t, 122
  - pd\_packet\_t, 123
- sop\_dp
  - pd\_do\_t::ACT\_CBL\_VDO\_1, 25
  - pd\_do\_t::ACT\_CBL\_VDO, 22
  - pd\_do\_t::STD\_CBL\_VDO, 171
- sop\_state
  - alt\_mode\_info\_t, 36
- sop\_t
  - pd.h, 400
- sop\_type
  - vdm\_msg\_info\_t, 193
- source\_only
  - pd\_stack\_conf\_t, 139
- spec\_rev
  - pd\_hdr\_t::PD\_HDR, 120
- spec\_rev\_cbl
  - dpm\_status\_t, 86
- spec\_rev\_peer
  - dpm\_status\_t, 86
- spec\_rev\_sop\_live
  - dpm\_status\_t, 86
- spec\_rev\_sop\_prime\_live
  - dpm\_status\_t, 86
- spm\_cfg\_tbl\_offset
  - pd\_port\_config\_t, 132
- sr\_async\_thresh
  - pwr\_params\_t, 149
- sr\_enable
  - pwr\_params\_t, 149
- sr\_fall\_time
  - pwr\_params\_t, 149
- sr\_rise\_time
  - pwr\_params\_t, 149
- sr\_supply\_doubler
  - pwr\_params\_t, 149
- src\_cap\_p
  - dpm\_status\_t, 86
- src\_cap\_start\_delay
  - dpm\_status\_t, 87
- src\_cur\_level
  - dpm\_status\_t, 87
- src\_cur\_level\_live
  - dpm\_status\_t, 87
- src\_cur\_rdo
  - dpm\_status\_t, 87
- src\_gate\_drv\_str
  - pwr\_params\_t, 149
- src\_gen
  - pd\_do\_t, 116
- src\_input\_change
  - pd\_do\_t::ADO\_ALERT, 29
- src\_last\_rdo
  - dpm\_status\_t, 87
- src\_pdo
  - dpm\_status\_t, 87
- src\_pdo\_cnt
  - pd\_port\_config\_t, 132
- src\_pdo\_count
  - dpm\_status\_t, 87
- src\_pdo\_list
  - pd\_port\_config\_t, 132
- src\_pdo\_mask
  - dpm\_status\_t, 87
- src\_period
  - dpm\_status\_t, 87
- src\_rdo
  - dpm\_status\_t, 88
- src\_sel
  - chg\_cfg\_params\_t, 64
- src\_sel\_pdo

- dpm\_status\_t, 88
- srom.h
  - CALL\_MAP, 545
- ss\_lanes
  - pd\_do\_t::ACT\_CBL\_VDO\_2, 27
- ss\_supp
  - pd\_do\_t::ACT\_CBL\_VDO\_2, 27
- ssrx1
  - pd\_do\_t::STD\_AMA\_VDO, 168
  - pd\_do\_t::STD\_CBL\_VDO, 171
- ssrx2
  - pd\_do\_t::STD\_AMA\_VDO, 168
  - pd\_do\_t::STD\_CBL\_VDO, 171
- sstx1
  - pd\_do\_t::STD\_AMA\_VDO, 168
  - pd\_do\_t::STD\_CBL\_VDO, 171
- sstx2
  - pd\_do\_t::STD\_AMA\_VDO, 168
  - pd\_do\_t::STD\_CBL\_VDO, 171
- st\_ver
  - pd\_do\_t::STD\_VDM\_HDR, 176
- state
  - bc\_dp\_dm\_state\_t, 57
  - cc\_state\_t, 61
- status
  - fw\_img\_status\_t, 96
  - pd\_contract\_info\_t, 111
  - pd\_port\_status\_t, 137
- status.h
  - CCG\_STATUS\_CODE\_OFFSET, 546
  - CCG\_STATUS\_TO\_HPI\_RESPONSE, 546
  - ccg\_status\_t, 546
- std\_ama\_vdo
  - pd\_do\_t, 116
- std\_ama\_vdo\_pd3
  - pd\_do\_t, 116
- std\_cbl\_vdo
  - pd\_do\_t, 116
- std\_cert\_vdo
  - pd\_do\_t, 116
- std\_dp\_vdo
  - pd\_do\_t, 116
- std\_id\_hdr
  - pd\_do\_t, 116
- std\_prod\_vdo
  - pd\_do\_t, 117
- std\_svid\_res
  - pd\_do\_t, 117
- std\_vdm\_cmd\_t
  - pd.h, 400
- std\_vdm\_cmd\_type\_t
  - pd.h, 400
- std\_vdm\_hdr
  - pd\_do\_t, 117
- std\_vdm\_prod\_t
  - pd.h, 401
- std\_vdm\_ver\_t
  - pd.h, 401
- supply\_type
  - pd\_do\_t::BAT\_SNK, 54
  - pd\_do\_t::BAT\_SRC, 55
  - pd\_do\_t::FIXED\_SNK, 94
  - pd\_do\_t::FIXED\_SRC, 95
  - pd\_do\_t::PPS\_SNK, 144
  - pd\_do\_t::PPS\_SRC, 145
  - pd\_do\_t::SRC\_GEN, 167
  - pd\_do\_t::VAR\_SNK, 191
  - pd\_do\_t::VAR\_SRC, 192
- svid
  - alt\_mode\_evt\_t::ALT\_MODE\_EVT, 31
  - pd\_do\_t::STD\_VDM\_HDR, 176
  - pd\_do\_t::USTD\_QC\_4\_0\_HDR, 188
  - pd\_do\_t::USTD\_VDM\_HDR, 189
  - reg\_am\_t, 164
- svid\_emca\_vdo
  - alt\_mode\_reg\_info\_t, 38
- svid\_n
  - pd\_do\_t::STD\_SVID\_RESP\_VDO, 175
- svid\_n1
  - pd\_do\_t::STD\_SVID\_RESP\_VDO, 175
- svid\_vdm\_length
  - pd\_port\_config\_t, 132
- svid\_vdm\_offset
  - pd\_port\_config\_t, 132
- svid\_vdo
  - alt\_mode\_reg\_info\_t, 38
- swap.h
  - eval\_dr\_swap, 310
  - eval\_fr\_swap, 310
  - eval\_pr\_swap, 310
  - eval\_vconn\_swap, 311
- swap\_response
  - dpm\_status\_t, 88
  - pd\_port\_config\_t, 132
- sys\_fw\_metadata\_t, 178
  - active\_boot\_app, 179
  - boot\_app\_id, 179
  - boot\_app\_ver\_status, 179
  - boot\_app\_version, 179
  - boot\_last\_row, 179
  - boot\_seq, 179
  - fw\_checksum, 179
  - fw\_entry, 179
  - fw\_size, 179
  - fw\_version, 180
  - metadata\_valid, 180
  - reserved1, 180
  - reserved2, 180
- sys\_fw\_mode\_t
  - system.h, 550
- sys\_get\_bcdDevice\_version
  - system.h, 551
- sys\_get\_boot\_version
  - system.h, 551
- sys\_get\_custom\_info\_addr
  - system.h, 551

- sys\_get\_device\_mode
  - system.h, 551
- sys\_get\_fw\_img1\_start\_addr
  - system.h, 552
- sys\_get\_fw\_img2\_start\_addr
  - system.h, 552
- sys\_get\_img1\_fw\_version
  - system.h, 552
- sys\_get\_img2\_fw\_version
  - system.h, 552
- sys\_get\_recent\_fw\_image
  - system.h, 553
- sys\_get\_silicon\_id
  - system.h, 553
- sys\_hw\_error\_type\_t
  - app.h, 244
- sys\_pwr
  - auto\_cfg\_settings\_t, 51
- sys\_set\_device\_mode
  - system.h, 553
- system.h
  - get\_silicon\_revision, 550
  - SYS\_APP\_VERSION\_OFFSET, 548
  - SYS\_BOOT\_MODE\_RQT\_SIG, 548
  - SYS\_BOOT\_TYPE\_APP\_PRIORITY\_POS, 548
  - SYS\_BOOT\_TYPE\_FIELD\_OFFSET, 548
  - SYS\_BOOT\_TYPE\_FW\_UPDATE\_INTERFACE↔  
\_MASK, 549
  - SYS\_BOOT\_TYPE\_FW\_UPDATE\_INTERFACE↔  
\_POS, 549
  - SYS\_BOOT\_TYPE\_SECURE\_BOOT\_MASK, 549
  - SYS\_BOOT\_VERSION\_ADDRESS, 549
  - SYS\_CONFIG\_TABLE\_SIGN, 549
  - SYS\_FW\_CUSTOM\_INFO\_OFFSET, 549
  - SYS\_FW\_VERSION\_OFFSET, 549
  - SYS\_INVALID\_FW\_START\_ADDR, 549
  - SYS\_METADATA\_VALID\_SIG, 549
  - SYS\_PSEUDO\_METADATA\_VALID\_SIG, 550
  - SYS\_SILICON\_ID\_MASK, 550
  - SYS\_SILICON\_ID\_OFFSET, 550
  - sys\_fw\_mode\_t, 550
  - sys\_get\_bcdDevice\_version, 551
  - sys\_get\_boot\_version, 551
  - sys\_get\_custom\_info\_addr, 551
  - sys\_get\_device\_mode, 551
  - sys\_get\_fw\_img1\_start\_addr, 552
  - sys\_get\_fw\_img2\_start\_addr, 552
  - sys\_get\_img1\_fw\_version, 552
  - sys\_get\_img2\_fw\_version, 552
  - sys\_get\_recent\_fw\_image, 553
  - sys\_get\_silicon\_id, 553
  - sys\_set\_device\_mode, 553
- system/boot.h, 518
- system/ccgx\_host\_sdk\_desc.h, 525
- system/ccgx\_version.h, 525
- system/flash.h, 526
- system/gpio.h, 532
- system/instrumentation.h, 544
- system/srom.h, 545
- system/status.h, 546
- system/system.h, 547
- system/timer.h, 554
- system/timer\_id.h, 560
- system/utils.h, 566
- system\_connect\_ovp\_trip
  - hal\_ccgx.h, 428
- system\_disconnect\_ovp\_trip
  - hal\_ccgx.h, 428
- system\_init
  - hal\_ccgx.h, 429
- system\_sleep
  - app.h, 257
- system\_vbus\_ocp\_dis
  - hal\_ccgx.h, 429
- system\_vbus\_ocp\_en
  - hal\_ccgx.h, 429
- system\_vbus\_rcp\_dis
  - hal\_ccgx.h, 429
- system\_vbus\_rcp\_en
  - hal\_ccgx.h, 430
- system\_vbus\_scp\_dis
  - hal\_ccgx.h, 430
- system\_vbus\_scp\_en
  - hal\_ccgx.h, 430
- system\_vconn\_ocp\_dis
  - app.h, 257
- system\_vconn\_ocp\_en
  - app.h, 258
- TBT\_ALT\_MODE\_ID
  - intel\_vid.h, 294
- TBT\_CTRLR\_ALPINE\_RIDGE\_DUAL
  - pd.h, 378
- TBT\_CTRLR\_ALPINE\_RIDGE\_SGL
  - pd.h, 379
- TBT\_CTRLR\_TITAN\_RIDGE\_DUAL
  - pd.h, 379
- TBT\_CTRLR\_TITAN\_RIDGE\_SGL
  - pd.h, 379
- TBT\_EXIT
  - intel\_vid.h, 294
- TBT\_GEN\_3
  - pd.h, 379
- TBT\_MODE\_EXIT\_CHECK\_PERIOD
  - app.h, 242
- TBT\_SVID
  - pd.h, 379
- TBT\_VDO\_IDX
  - intel\_vid.h, 294
- THROTTLE\_DEBOUNCE\_PERIOD
  - app.h, 242
- THROTTLE\_WAIT\_FOR\_PD\_PERIOD
  - app.h, 242
- TIMER\_INVALID\_INDEX
  - timer.h, 555
- TIMER\_INVALID\_ID
  - timer.h, 555

TIMER\_MAX\_TIMEOUT  
     timer.h, 555  
 TIMER\_NUM\_TIMERS  
     timer.h, 555  
 TYPE\_A\_CUR\_SENSE\_TIMER\_PERIOD  
     app.h, 242  
 TYPE\_A\_PWM\_STEP\_TIMER\_PERIOD  
     app.h, 242  
 TYPE\_A\_REG\_SWITCH\_TIMER\_PERIOD  
     app.h, 242  
 TYPEC\_ATTACH\_WAIT\_ENTRY\_DELAY\_PERIOD  
     pd.h, 379  
 TYPEC\_CC\_DEBOUNCE\_TIMER\_PERIOD  
     pd.h, 379  
 TYPEC\_DRP\_TRY\_TIMER\_PERIOD  
     pd.h, 379  
 TYPEC\_ERROR\_RECOVERY\_TIMER\_PERIOD  
     pd.h, 379  
 TYPEC\_FSM\_GENERIC  
     pd.h, 380  
 TYPEC\_FSM\_NONE  
     pd.h, 380  
 TYPEC\_PD3\_RPCHANGE\_DEBOUNCE\_PERIOD  
     pd.h, 380  
 TYPEC\_PD\_DEBOUNCE\_TIMER\_PERIOD  
     pd.h, 380  
 TYPEC\_RD\_DEBOUNCE\_TIMER\_PERIOD  
     pd.h, 380  
 TYPEC\_SRC\_DETACH\_DEBOUNCE\_PERIOD  
     pd.h, 380  
 TYPEC\_TRY\_TIMEOUT\_PERIOD  
     pd.h, 380  
 table  
     pasc\_valley\_table\_t, 107  
 table\_checksum  
     pd\_config\_t, 110  
 table\_len  
     alt\_cfg\_settings\_t, 30  
     auto\_cfg\_settings\_t, 51  
     bat\_chg\_params\_t, 52  
     bb\_settings\_t, 57  
     chg\_cfg\_params\_t, 64  
     custom\_alt\_cfg\_settings\_t, 66  
     custom\_host\_cfg\_settings\_t, 67  
     icl\_tgl\_cfg\_settings\_t, 100  
     ocp\_settings\_t, 101  
     otp\_settings\_t, 103  
     ovp\_settings\_t, 104  
     pwr\_params\_t, 149  
     rcp\_settings\_t, 152  
     scp\_settings\_t, 165  
     tbthost\_cfg\_settings\_t, 185  
     uvp\_settings\_t, 190  
     vconn\_ocp\_settings\_t, 193  
 table\_sign  
     pd\_config\_t, 110  
 table\_size  
     pd\_config\_t, 110  
 table\_type  
     pd\_config\_t, 110  
 table\_version  
     pd\_config\_t, 110  
 tbt\_cbl\_gen\_t  
     intel\_vid.h, 295  
 tbt\_cbl\_speed\_t  
     intel\_vid.h, 295  
 tbt\_cbl\_vdo  
     pd\_do\_t, 117  
 tbt\_ctrlr\_type  
     tbthost\_cfg\_settings\_t, 186  
 tbt\_state\_t  
     intel\_vid.h, 295  
 tbt\_ufp\_vdo  
     pd\_do\_t, 117  
 tbt\_vdo  
     pd\_do\_t, 117  
 tbthost\_cfg\_settings\_t, 184  
     host\_support, 185  
     hpd\_handling, 185  
     non\_tbt\_mux, 185  
     pref\_data\_role, 185  
     pref\_pwr\_role, 185  
     rsvd0, 185  
     sbu\_config, 185  
     table\_len, 185  
     tbt\_ctrlr\_type, 186  
     usb3\_support, 186  
     usb4\_support, 186  
     vpro\_capable, 186  
 tbthost\_cfg\_tbl\_offset  
     pd\_port\_config\_t, 132  
 temp\_status  
     pd\_power\_status\_t, 138  
 tgt\_id\_header  
     atch\_tgt\_info\_t, 49  
 tgt\_svid  
     atch\_tgt\_info\_t, 50  
 therm\_1\_enable  
     otp\_settings\_t, 103  
 therm\_type  
     otp\_settings\_t, 103  
 therm\_type\_1  
     otp\_settings\_t, 103  
 threshold  
     ocp\_settings\_t, 102  
     ovp\_settings\_t, 104  
     scp\_settings\_t, 165  
     uvp\_settings\_t, 190  
     vconn\_ocp\_settings\_t, 193  
 threshold2  
     ocp\_settings\_t, 102  
 timeout  
     dpm\_pd\_cmd\_buf\_t, 72  
 timer.h  
     TIMER\_INVALID\_INDEX, 555  
     TIMER\_INVALID\_ID, 555



- TIMER\_MAX\_TIMEOUT, 555
- TIMER\_NUM\_TIMERS, 555
- timer\_cb\_t, 555
- timer\_enter\_sleep, 556
- timer\_get\_count, 556
- timer\_get\_multiplier, 556
- timer\_id\_t, 555
- timer\_init, 557
- timer\_is\_running, 557
- timer\_num\_active, 557
- timer\_range\_enabled, 558
- timer\_start, 558
- timer\_start\_wocb, 558
- timer\_stop, 559
- timer\_stop\_all, 559
- timer\_stop\_range, 560
- timer\_cb\_t
  - timer.h, 555
- timer\_enter\_sleep
  - timer.h, 556
- timer\_get\_count
  - timer.h, 556
- timer\_get\_multiplier
  - timer.h, 556
- timer\_id.h
  - ccg\_timer\_id\_t, 561
- timer\_id\_t
  - timer.h, 555
- timer\_init
  - timer.h, 557
- timer\_is\_running
  - timer.h, 557
- timer\_num\_active
  - timer.h, 557
- timer\_range\_enabled
  - timer.h, 558
- timer\_start
  - timer.h, 558
- timer\_start\_wocb
  - timer.h, 558
- timer\_stop
  - timer.h, 559
- timer\_stop\_all
  - timer.h, 559
- timer\_stop\_range
  - timer.h, 560
- toggle
  - dpm\_status\_t, 88
- tr\_hpd\_deinit
  - intel\_ridge.h, 291
- tr\_hpd\_init
  - intel\_ridge.h, 292
- tr\_hpd\_sendevt
  - intel\_ridge.h, 292
- tr\_is\_hpd\_change
  - intel\_ridge.h, 292
- tr\_msg\_id
  - prl\_cntrs\_t, 146
- trig\_cbl\_rst
  - app\_status\_t, 47
- try\_src\_en
  - pd\_port\_config\_t, 132
- try\_src\_snk
  - dpm\_status\_t, 88
- try\_src\_snk\_dis
  - dpm\_status\_t, 88
- try\_src\_snk\_t
  - pd.h, 401
- turn\_off\_temp\_limit
  - app\_status\_t, 47
- turn\_on\_temp\_limit
  - app\_status\_t, 47
- tx\_buf
  - pd\_status\_t, 142
- tx\_busy
  - pd\_status\_t, 142
- tx\_count
  - pd\_status\_t, 142
- tx\_extd
  - pd\_status\_t, 142
- tx\_extd\_hdr
  - pd\_status\_t, 142
- tx\_header
  - pd\_status\_t, 142
- tx\_msg\_type
  - pd\_status\_t, 142
- tx\_sop
  - pd\_status\_t, 142
- type\_a\_chg\_tbl\_offset
  - pd\_port\_config\_t, 133
- type\_a\_enable
  - pd\_port\_config\_t, 133
- type\_a\_pwr\_tbl\_offset
  - pd\_port\_config\_t, 133
- typec\_abc
  - pd\_do\_t::ACT\_CBL\_VDO\_1, 25
  - pd\_do\_t::ACT\_CBL\_VDO, 22
  - pd\_do\_t::PAS\_CBL\_VDO, 106
  - pd\_do\_t::STD\_CBL\_VDO, 172
- typec\_assert\_rd
  - typec\_manager.h, 425
- typec\_assert\_rp
  - typec\_manager.h, 425
- typec\_change\_rp
  - typec\_manager.h, 425
- typec\_deepsleep
  - typec\_manager.h, 426
- typec\_evt
  - dpm\_status\_t, 88
- typec\_fsm
  - typec\_manager.h, 426
- typec\_fsm\_state
  - dpm\_status\_t, 88
- typec\_fsm\_state\_t
  - pd.h, 402
- typec\_init

- typec\_manager.h, 426
- typec\_is\_busy
  - typec\_manager.h, 426
- typec\_manager.h
  - typec\_assert\_rd, 425
  - typec\_assert\_rp, 425
  - typec\_change\_rp, 425
  - typec\_deepsleep, 426
  - typec\_fsm, 426
  - typec\_init, 426
  - typec\_is\_busy, 426
  - typec\_start, 427
  - typec\_stop, 427
- typec\_plug
  - pd\_do\_t::ACT\_CBL\_VDO, 22
  - pd\_do\_t::PAS\_CBL\_VDO, 106
  - pd\_do\_t::STD\_CBL\_VDO, 172
- typec\_start
  - typec\_manager.h, 427
- typec\_stop
  - typec\_manager.h, 427
- u3\_power
  - pd\_do\_t::ACT\_CBL\_VDO\_2, 27
- u3\_u0\_trans
  - pd\_do\_t::ACT\_CBL\_VDO\_2, 27
- UCSI\_BUFFER\_SIZE
  - ucsi.h, 579
- UCSI\_CONNECT\_EVENT\_PERIOD
  - app.h, 242
- UCSI\_NOTIFICATION\_EN\_ALL
  - ucsi.h, 579
- UCSI\_NOTIFICATION\_EN\_REQ
  - ucsi.h, 580
- UCSI\_READ\_PENDING\_EVENT
  - ucsi.h, 580
- UCSI\_READ\_PENDING\_MASK
  - ucsi.h, 580
- UCSI\_SET\_RP\_1\_5A
  - ucsi.h, 580
- UCSI\_SET\_RP\_3A
  - ucsi.h, 580
- UCSI\_SET\_RP\_DEF
  - ucsi.h, 580
- UCSI\_SET\_RP\_PPM\_DEFAULT
  - ucsi.h, 580
- UFP\_ALT\_MODE\_HPI\_MASK
  - alt\_modes\_mngr.h, 206
- UFP\_D\_CONN
  - dp\_sid.h, 223
- UFP\_NON\_PH\_ALT\_MODE\_SUPP\_MASK
  - pd.h, 380
- UFP\_VDO\_1\_RECFG\_ALT\_MODE\_PARAM\_MASK
  - pd.h, 380
- USB2\_ENABLE
  - intel\_vid.h, 294
- USB4\_EUDO\_HOST\_PARAM\_SHIFT
  - vdm\_task\_mngr.h, 227
- USB\_CONFIG\_SELECT
  - dp\_sid.h, 223
- ucsi.h
  - ALT\_MODES\_RECIPIENT\_CONNECTOR, 577
  - ALT\_MODES\_RECIPIENT\_SOP\_DPRIME, 577
  - ALT\_MODES\_RECIPIENT\_SOP\_PRIME, 577
  - ALT\_MODES\_RECIPIENT\_SOP, 577
  - CABLE\_CURR\_3A, 577
  - CABLE\_CURR\_5A, 578
  - CABLE\_CURR\_DFLT, 578
  - CABLE\_SPEED\_10GBPS, 578
  - CABLE\_SPEED\_480MBPS, 578
  - CABLE\_SPEED\_5GBPS, 578
  - CABLE\_VDO\_DIRECTION, 578
  - is\_port\_connect\_changed, 581
  - MAX\_DPM\_CMD\_RETRY\_COUNT, 578
  - POM\_BC, 578
  - POM\_CUR\_LEVEL\_1\_5A, 578
  - POM\_CUR\_LEVEL\_3A, 579
  - POM\_CUR\_LEVEL\_DEF, 579
  - POM\_NO\_CONSUMER, 579
  - POM\_PD, 579
  - SRC\_CUR\_LEVEL\_1\_5A, 579
  - SRC\_CUR\_LEVEL\_3A, 579
  - SRC\_CUR\_LEVEL\_DEF, 579
  - UCSI\_BUFFER\_SIZE, 579
  - UCSI\_NOTIFICATION\_EN\_ALL, 579
  - UCSI\_NOTIFICATION\_EN\_REQ, 580
  - UCSI\_READ\_PENDING\_EVENT, 580
  - UCSI\_READ\_PENDING\_MASK, 580
  - UCSI\_SET\_RP\_1\_5A, 580
  - UCSI\_SET\_RP\_3A, 580
  - UCSI\_SET\_RP\_DEF, 580
  - UCSI\_SET\_RP\_PPM\_DEFAULT, 580
  - ucsi\_configure\_send\_vdm, 581
  - ucsi\_init, 581
  - ucsi\_notify, 582
  - ucsi\_pd\_event\_handler, 582
  - ucsi\_refresh\_conn\_cnt, 582
  - ucsi\_sleep\_allowed, 583
  - ucsi\_task, 583
  - VDM\_DISCOVER\_ID, 580
  - VDM\_DISCOVER\_MODES, 580
  - VDM\_DISCOVER\_SVID, 581
- ucsi/ucsi.h, 576
- ucsi\_clear\_status\_bit
  - hpi.h, 328
- ucsi\_configure\_send\_vdm
  - ucsi.h, 581
- ucsi\_get\_status\_bit
  - hpi.h, 328
- ucsi\_init
  - ucsi.h, 581
- ucsi\_notify
  - ucsi.h, 582
- ucsi\_pd\_event\_handler
  - ucsi.h, 582
- ucsi\_refresh\_conn\_cnt
  - ucsi.h, 582

- ucsi\_reg\_reset
  - hpi.h, 328
- ucsi\_set\_status\_bit
  - hpi.h, 328
- ucsi\_sleep\_allowed
  - ucsi.h, 583
- ucsi\_task
  - ucsi.h, 583
- ufp\_alt\_mode\_mask
  - app\_status\_t, 47
- ufp\_asgmt
  - pd\_do\_t::DP\_CONFIG\_VDO, 70
- ufp\_d\_pin
  - pd\_do\_t::STD\_DP\_VDO, 174
- ufp\_mask
  - alt\_cfg\_settings\_t, 30
- ufp\_vdo\_1
  - pd\_do\_t, 117
- unchunk\_sup
  - pd\_do\_t::FIXED\_SRC, 95
  - pd\_do\_t::RDO\_BAT\_GIVEBACK, 155
  - pd\_do\_t::RDO\_BAT, 153
  - pd\_do\_t::RDO\_FIXED\_VAR\_GIVEBACK, 158
  - pd\_do\_t::RDO\_FIXED\_VAR, 156
  - pd\_do\_t::RDO\_GEN\_GVB, 161
  - pd\_do\_t::RDO\_GEN, 160
  - pd\_do\_t::RDO\_PPS, 163
- unchunk\_sup\_live
  - dpm\_status\_t, 89
- unchunk\_sup\_peer
  - dpm\_status\_t, 89
- unconstrained\_pwr\_en
  - auto\_cfg\_settings\_t, 51
- usb2\_0
  - pd\_do\_t::STD\_DP\_VDO, 174
- usb2\_hub\_hops
  - pd\_do\_t::ACT\_CBL\_VDO\_2, 27
- usb2\_supp
  - app\_status\_t, 48
  - pd\_do\_t::ACT\_CBL\_VDO\_2, 27
- usb3\_drd
  - pd\_do\_t::ENTERUSB\_VDO, 91
- usb3\_supp
  - app\_status\_t, 48
- usb3\_support
  - tbthost\_cfg\_settings\_t, 186
- usb4\_active
  - app\_status\_t, 48
- usb4\_data\_rst\_cnt
  - app\_status\_t, 48
- usb4\_drd
  - pd\_do\_t::ENTERUSB\_VDO, 91
- usb4\_en
  - dpm\_status\_t, 89
- usb4\_flag\_t
  - vdm\_task\_mgr.h, 227
- usb4\_supp
  - pd\_do\_t::ACT\_CBL\_VDO\_2, 27
- usb4\_support
  - tbthost\_cfg\_settings\_t, 186
- usb4\_update\_data\_status
  - vdm\_task\_mgr.h, 230
- usb\_cfg
  - pd\_do\_t::DP\_STATUS\_VDO, 71
- usb\_comm\_cap
  - pd\_do\_t::FIXED\_SNK, 94
  - pd\_do\_t::FIXED\_SRC, 95
  - pd\_do\_t::RDO\_BAT\_GIVEBACK, 155
  - pd\_do\_t::RDO\_BAT, 153
  - pd\_do\_t::RDO\_FIXED\_VAR\_GIVEBACK, 158
  - pd\_do\_t::RDO\_FIXED\_VAR, 157
  - pd\_do\_t::RDO\_GEN\_GVB, 162
  - pd\_do\_t::RDO\_GEN, 160
  - pd\_do\_t::RDO\_PPS, 163
- usb\_data\_sig\_t
  - pd.h, 402
- usb\_dev
  - pd\_do\_t::STD\_VDM\_ID\_HDR, 178
- usb\_dev\_cap\_t
  - pd.h, 402
- usb\_gen
  - pd\_do\_t::ACT\_CBL\_VDO\_2, 27
- usb\_host
  - pd\_do\_t::STD\_VDM\_ID\_HDR, 178
- usb\_host\_cap\_t
  - pd.h, 403
- usb\_mode
  - pd\_do\_t::ENTERUSB\_VDO, 91
- usb\_pid
  - pd\_do\_t::STD\_PROD\_VDO, 174
- usb\_role\_t
  - pd.h, 403
- usb\_sig
  - pd\_do\_t::UFP\_VDO\_1, 187
- usb\_sig\_supp\_t
  - pd.h, 403
- usb\_ss\_sup
  - pd\_do\_t::ACT\_CBL\_VDO\_1, 25
  - pd\_do\_t::ACT\_CBL\_VDO, 22
  - pd\_do\_t::PAS\_CBL\_VDO, 106
  - pd\_do\_t::STD\_AMA\_VDO\_PD3, 169
  - pd\_do\_t::STD\_AMA\_VDO, 168
  - pd\_do\_t::STD\_CBL\_VDO, 172
- usb\_suspend\_sup
  - pd\_do\_t::FIXED\_SRC, 95
- usb\_vid
  - pd\_do\_t::STD\_VDM\_ID\_HDR, 178
- usb\_xid
  - pd\_do\_t::STD\_CERT\_VDO, 173
- ustd\_qc\_4\_0\_hdr
  - pd\_do\_t, 117
- ustd\_vdm\_hdr
  - pd\_do\_t, 117
- utils.h
  - apply\_threshold, 571
  - BUSY\_WAIT\_US, 567

- BYTE\_GET\_LOWER\_NIBBLE, 567
- BYTE\_GET\_UPPER\_NIBBLE, 567
- crc16, 571
- DIV\_ROUND\_NEAREST, 567
- DIV\_ROUND\_UP, 567
- DWORD\_GET\_BYTE0, 568
- DWORD\_GET\_BYTE1, 568
- DWORD\_GET\_BYTE2, 568
- DWORD\_GET\_BYTE3, 568
- div\_round\_up, 571
- event\_group\_clear\_event, 572
- event\_group\_clear\_events\_by\_val, 572
- event\_group\_get\_event, 572
- event\_group\_set\_event, 573
- event\_group\_set\_events\_by\_val, 573
- GET\_MAX, 568
- GET\_MIN, 568
- MAKE\_DWORD\_FROM\_WORD, 569
- MAKE\_DWORD, 568
- MAKE\_WORD, 569
- MEM\_CMP, 569
- MEM\_COPY, 569
- MEM\_SET, 569
- mem\_calculate\_byte\_checksum, 573
- mem\_calculate\_dword\_checksum, 574
- mem\_calculate\_word\_checksum, 574
- mem\_copy, 575
- mem\_copy\_word, 575
- mem\_set, 575
- REG\_FIELD\_GET, 569
- REG\_FIELD\_UPDATE, 570
- REV\_BYTE\_ORDER, 570
- WORD\_GET\_LSB, 570
- WORD\_GET\_MSB, 570
- uvdm\_supp
  - alt\_mode\_info\_t, 36
- uvp\_settings\_t, 189
  - debounce, 190
  - retry\_cnt, 190
  - table\_len, 190
  - threshold, 190
- uvp\_tbl\_offset
  - pd\_port\_config\_t, 133
- v0
  - pasc\_valley\_table\_t, 108
- v1
  - pasc\_valley\_table\_t, 108
- v2
  - pasc\_valley\_table\_t, 108
- VBUS\_OCP\_GPIO\_ACTIVE\_HIGH
  - pdss\_hal.h, 446
- VBUS\_OCP\_GPIO\_ACTIVE\_LOW
  - pdss\_hal.h, 446
- VBUS\_OCP\_MODE\_EXT
  - pdss\_hal.h, 446
- VBUS\_OCP\_MODE\_INT\_AUTOCTRL
  - pdss\_hal.h, 446
- VBUS\_OCP\_MODE\_INT\_SW\_DB
  - pdss\_hal.h, 446
- VBUS\_OCP\_MODE\_INT
  - pdss\_hal.h, 446
- VBUS\_OCP\_MODE\_POLLING
  - pdss\_hal.h, 447
- VDM\_DISCOVER\_ID
  - ucsi.h, 580
- VDM\_DISCOVER\_MODES
  - ucsi.h, 580
- VDM\_DISCOVER\_SVID
  - ucsi.h, 581
- VDM\_HDR
  - alt\_modes\_mngr.h, 206
- VDM\_HEADER\_IDX
  - pd.h, 381
- VDO\_START\_IDX
  - alt\_modes\_mngr.h, 206
- VPRO\_ALT\_MODE\_ID
  - intel\_vid.h, 294
- VSAFE\_0V\_HARD\_RESET
  - pd.h, 381
- VSAFE\_0V\_PR\_SWAP\_SNK\_SRC
  - pd.h, 381
- VSAFE\_0V
  - pd.h, 381
- VSAFE\_12V
  - pd.h, 381
- VSAFE\_13V
  - pd.h, 381
- VSAFE\_15V
  - pd.h, 381
- VSAFE\_19V
  - pd.h, 381
- VSAFE\_20V
  - pd.h, 381
- VSAFE\_3\_6V
  - pd.h, 382
- VSAFE\_5V
  - pd.h, 382
- VSAFE\_9V
  - pd.h, 382
- val
  - alt\_mode\_evt\_t, 32
  - alt\_mode\_hw\_evt\_t, 34
  - fw\_img\_status\_t, 96
  - pd\_do\_t, 118
  - pd\_extd\_hdr\_t, 118
  - pd\_hdr\_t, 121
  - pd\_port\_status\_t, 137
- var\_snk
  - pd\_do\_t, 118
- var\_src
  - pd\_do\_t, 118
- vbatt\_cutoff\_volt
  - bat\_chg\_params\_t, 52
- vbatt\_dischg\_en\_volt
  - bat\_chg\_params\_t, 53
- vbatt\_max\_cur

- bat\_chg\_params\_t, 53
- vbatt\_max\_volt
  - bat\_chg\_params\_t, 53
- vbtr\_down\_step\_width
  - pwr\_params\_t, 149
- vbtr\_up\_step\_width
  - pwr\_params\_t, 150
- vbus\_cf\_cbk\_t
  - pdss\_hal.h, 448
- vbus\_cur
  - pd\_do\_t::ACT\_CBL\_VDO\_1, 25
  - pd\_do\_t::ACT\_CBL\_VDO, 23
  - pd\_do\_t::PAS\_CBL\_VDO, 106
  - pd\_do\_t::STD\_CBL\_VDO, 172
- vbus\_dflt\_volt
  - pwr\_params\_t, 150
- vbus\_discharge\_off
  - app.h, 258
  - app\_cbk\_t, 41
- vbus\_discharge\_on
  - app.h, 258
  - app\_cbk\_t, 41
- vbus\_get\_value
  - app.h, 259
  - app\_cbk\_t, 41
- vbus\_is\_present
  - app.h, 259
  - app\_cbk\_t, 41
- vbus\_load\_chg\_cbk\_t
  - pdss\_hal.h, 449
- vbus\_max\_volt
  - pwr\_params\_t, 150
- vbus\_min\_volt
  - pwr\_params\_t, 150
- vbus\_ocp\_handler
  - hal\_ccgx.h, 431
- vbus\_offset\_volt
  - pwr\_params\_t, 150
- vbus\_ovp\_mode\_t
  - pdss\_hal.h, 455
- vbus\_rcp\_handler
  - hal\_ccgx.h, 431
- vbus\_req
  - pd\_do\_t::STD\_AMA\_VDO\_PD3, 169
  - pd\_do\_t::STD\_AMA\_VDO, 168
- vbus\_scp\_handler
  - hal\_ccgx.h, 431
- vbus\_thru\_cbl
  - pd\_do\_t::ACT\_CBL\_VDO\_1, 25
  - pd\_do\_t::ACT\_CBL\_VDO, 23
  - pd\_do\_t::STD\_CBL\_VDO, 172
- vbus\_uvp\_mode\_t
  - pdss\_hal.h, 455
- vcon\_pwr
  - pd\_do\_t::STD\_AMA\_VDO\_PD3, 169
  - pd\_do\_t::STD\_AMA\_VDO, 168
- vcon\_req
  - pd\_do\_t::STD\_AMA\_VDO\_PD3, 170
- pd\_do\_t::STD\_AMA\_VDO, 168
- vconn\_change\_handler
  - app.h, 259
- vconn\_disable
  - app.h, 261
  - app\_cbk\_t, 41
- vconn\_enable
  - app.h, 261
  - app\_cbk\_t, 41
- vconn\_is\_present
  - app.h, 261
  - app\_cbk\_t, 41
- vconn\_logical
  - dpm\_status\_t, 89
- vconn\_ocp\_settings\_t, 192
  - debounce, 192
  - retry\_cnt, 192
  - table\_len, 193
  - threshold, 193
- vconn\_ocp\_tbl\_offset
  - pd\_port\_config\_t, 133
- vconn\_on
  - pd\_port\_status\_t::PD\_PORT\_STAT, 136
- vconn\_retain
  - dpm\_status\_t, 89
  - pd\_port\_config\_t, 133
- vconn\_src
  - pd\_port\_status\_t::PD\_PORT\_STAT, 136
- vdm.h
  - eval\_enter\_usb, 311
  - eval\_vdm, 312
  - vdm\_data\_init, 312
  - vdm\_update\_data, 313
- vdm\_ams\_t
  - pd.h, 404
- vdm\_data\_init
  - vdm.h, 312
- vdm\_evt\_t
  - vdm\_task\_mgr.h, 228
- vdm\_get\_disc\_id\_resp
  - vdm\_task\_mgr.h, 230
- vdm\_get\_disc\_svid\_resp
  - vdm\_task\_mgr.h, 231
- vdm\_header
  - alt\_mode\_info\_t, 36
  - vdm\_msg\_info\_t, 193
- vdm\_msg\_info\_t, 193
  - sop\_type, 193
  - vdm\_header, 193
  - vdo, 193
  - vdo\_num, 194
- vdm\_prcs\_failed
  - app\_status\_t, 48
- vdm\_resp
  - app\_status\_t, 48
- vdm\_resp\_cbk
  - app\_status\_t, 48
- vdm\_resp\_cbk\_t

- pd.h, 384
- vdm\_resp\_t, 194
  - do\_count, 194
  - no\_resp, 194
  - resp\_buf, 194
- vdm\_retry\_pending
  - app\_status\_t, 48
- vdm\_task\_en
  - app\_status\_t, 48
- vdm\_task\_mgr
  - vdm\_task\_mgr.h, 231
- vdm\_task\_mgr.h
  - DATA\_RST\_RETRY\_NUMB, 227
  - enable\_vdm\_task\_mgr, 229
  - enter\_usb4, 229
  - is\_ufp\_disc\_started, 229
  - is\_vdm\_task\_idle, 230
  - MAX\_CABLE\_SVID\_SUPP, 227
  - MAX\_DISC\_SVID\_COUNT, 227
  - MAX\_SVID\_VDO\_SUPP, 227
  - modal\_op\_support, 232
  - PD\_DISC\_ID\_AMA\_VDO\_IDX, 227
  - PD\_SVID\_ID\_HDR\_VDO\_START\_IDX, 227
  - STD\_SVID, 227
  - USB4\_EUDO\_HOST\_PARAM\_SHIFT, 227
  - usb4\_flag\_t, 227
  - usb4\_update\_data\_status, 230
  - vdm\_evt\_t, 228
  - vdm\_get\_disc\_id\_resp, 230
  - vdm\_get\_disc\_svid\_resp, 231
  - vdm\_task\_mgr, 231
  - vdm\_task\_mgr\_deinit, 231
  - vdm\_task\_mgr\_exit\_modes, 232
  - vdm\_task\_t, 228
- vdm\_task\_mgr\_alt\_mode\_process
  - alt\_modes\_mgr.h, 216
- vdm\_task\_mgr\_deinit
  - vdm\_task\_mgr.h, 231
- vdm\_task\_mgr\_exit\_modes
  - vdm\_task\_mgr.h, 232
- vdm\_task\_t
  - vdm\_task\_mgr.h, 228
- vdm\_type
  - pd\_do\_t::STD\_VDM\_HDR, 176
  - pd\_do\_t::USTD\_QC\_4\_0\_HDR, 188
  - pd\_do\_t::USTD\_VDM\_HDR, 189
- vdm\_type\_t
  - pd.h, 404
- vdm\_update\_data
  - vdm.h, 313
- vdm\_ver
  - pd\_do\_t::USTD\_VDM\_HDR, 189
- vdm\_version
  - app\_status\_t, 49
- vdo
  - alt\_mode\_info\_t, 36
  - vdm\_msg\_info\_t, 193
- vdo\_max\_numb
  - alt\_mode\_info\_t, 36
- vdo\_numb
  - alt\_mode\_info\_t, 36
  - vdm\_msg\_info\_t, 194
- vdo\_version
  - pd\_do\_t::ACT\_CBL\_VDO\_1, 25
  - pd\_do\_t::ACT\_CBL\_VDO, 23
  - pd\_do\_t::DFP\_VDO, 68
  - pd\_do\_t::PAS\_CBL\_VDO, 106
  - pd\_do\_t::STD\_AMA\_VDO\_PD3, 170
  - pd\_do\_t::UFP\_VDO\_1, 187
- vin\_oc1
  - auto\_cfg\_settings\_t, 51
- vin\_oc2
  - auto\_cfg\_settings\_t, 51
- vin\_oc3
  - auto\_cfg\_settings\_t, 51
- vin\_throttling\_ctrl
  - auto\_cfg\_settings\_t, 51
- voltage
  - pd\_do\_t::FIXED\_SNK, 94
  - pd\_do\_t::FIXED\_SRC, 96
- vpro\_capable
  - tbthost\_cfg\_settings\_t, 186
- vpro\_dock\_host
  - pd\_do\_t::TBT\_VDO, 184
- vpro\_supp
  - pd\_do\_t::TBT\_UFP\_VDO, 182
- WORD\_GET\_LSB
  - utils.h, 570
- WORD\_GET\_MSB
  - utils.h, 570