

Vitesse API

Generated by Doxygen 1.8.14

Contents

1	Ethernet Virtual Connections	1
1.1	Port Configuration	1
1.2	Policer Configuration	1
1.3	EVCs	2
1.4	ECEs	2
1.5	EVC Statistics	3
1.6	ECE Statistics	3
2	Layer 2	5
2.1	MAC Address Table	5
2.2	Operational State	6
2.3	Spanning Tree	6
2.4	VLAN	6
2.5	VLAN Classification List	7
2.6	VLAN Translation	7
2.7	Port Isolation	8
2.8	Private VLAN	8
2.9	Asymmetric Private VLAN	8
2.10	Destination Port Groups	8
2.11	sFlow	9
2.12	Link Aggregation	9
2.13	Global Link Aggregation	9
2.14	Mirroring	10
2.15	Flooding Control	10
2.16	IPv4 Multicast	10
2.17	IPv6 Multicast	11
2.18	Ethernet Protection Switching	11
2.19	Ethernet Ring Protection Switching	11
2.20	VStaX	11

3	Layer 3	13
4	Security	15
4.1	Port Authentication (802.1X)	15
4.2	Access Control List	15
4.2.1	Access Control Entry	15
4.2.2	Port Configuration	16
4.2.3	Policer Configuration	16
5	Data Structure Index	17
5.1	Data Structures	17
6	File Index	29
6.1	File List	29
7	Data Structure Documentation	31
7.1	ib_par_cfg Struct Reference	31
7.1.1	Detailed Description	31
7.1.2	Field Documentation	31
7.1.2.1	value	31
7.1.2.2	min	32
7.1.2.3	max	32
7.2	port_custom_conf_t Struct Reference	32
7.2.1	Detailed Description	32
7.2.2	Field Documentation	33
7.2.2.1	enable	33
7.2.2.2	autoneg	33
7.2.2.3	fdx	33
7.2.2.4	flow_control	33
7.2.2.5	pfc	33
7.2.2.6	speed	34
7.2.2.7	dual_media_fiber_speed	34

7.2.2.8	max_length	34
7.2.2.9	exc_col_cont	34
7.2.2.10	adv_dis	34
7.2.2.11	max_tags	35
7.2.2.12	oper_up	35
7.2.2.13	frame_length_chk	35
7.3	serdes_fields_t Struct Reference	35
7.3.1	Detailed Description	35
7.3.2	Field Documentation	36
7.3.2.1	ob_post0	36
7.3.2.2	ob_sr	36
7.4	vtss_ace_frame_arp_t Struct Reference	36
7.4.1	Detailed Description	37
7.4.2	Field Documentation	37
7.4.2.1	smac	37
7.4.2.2	arp	37
7.4.2.3	req	37
7.4.2.4	unknown	37
7.4.2.5	smac_match	38
7.4.2.6	dmac_match	38
7.4.2.7	length	38
7.4.2.8	ip	38
7.4.2.9	ethernet	38
7.4.2.10	sip	39
7.4.2.11	dip	39
7.5	vtss_ace_frame_etype_t Struct Reference	39
7.5.1	Detailed Description	39
7.5.2	Field Documentation	39
7.5.2.1	dmac	40
7.5.2.2	smac	40

7.5.2.3	<code>etype</code>	40
7.5.2.4	<code>data</code>	40
7.6	<code>vtss_ace_frame_ipv4_t</code> Struct Reference	40
7.6.1	Detailed Description	41
7.6.2	Field Documentation	41
7.6.2.1	<code>ttl</code>	41
7.6.2.2	<code>fragment</code>	41
7.6.2.3	<code>options</code>	42
7.6.2.4	<code>ds</code>	42
7.6.2.5	<code>proto</code>	42
7.6.2.6	<code>sip</code>	42
7.6.2.7	<code>dip</code>	42
7.6.2.8	<code>data</code>	43
7.6.2.9	<code>sport</code>	43
7.6.2.10	<code>dport</code>	43
7.6.2.11	<code>tcp_fin</code>	43
7.6.2.12	<code>tcp_syn</code>	43
7.6.2.13	<code>tcp_rst</code>	44
7.6.2.14	<code>tcp_psh</code>	44
7.6.2.15	<code>tcp_ack</code>	44
7.6.2.16	<code>tcp_urg</code>	44
7.6.2.17	<code>sip_eq_dip</code>	44
7.6.2.18	<code>sport_eq_dport</code>	45
7.6.2.19	<code>seq_zero</code>	45
7.7	<code>vtss_ace_frame_ipv6_t</code> Struct Reference	45
7.7.1	Detailed Description	46
7.7.2	Field Documentation	46
7.7.2.1	<code>proto</code>	46
7.7.2.2	<code>sip</code>	46
7.7.2.3	<code>ttl</code>	46

7.7.2.4	ds	46
7.7.2.5	data	47
7.7.2.6	sport	47
7.7.2.7	dport	47
7.7.2.8	tcp_fin	47
7.7.2.9	tcp_syn	47
7.7.2.10	tcp_RST	48
7.7.2.11	tcp_psh	48
7.7.2.12	tcp_ack	48
7.7.2.13	tcp_urg	48
7.7.2.14	sip_eq_dip	48
7.7.2.15	sport_eq_dport	49
7.7.2.16	seq_zero	49
7.8	vtss_ace_frame_llc_t Struct Reference	49
7.8.1	Detailed Description	49
7.8.2	Field Documentation	49
7.8.2.1	dmac	50
7.8.2.2	smac	50
7.8.2.3	llc	50
7.9	vtss_ace_frame_snap_t Struct Reference	50
7.9.1	Detailed Description	50
7.9.2	Field Documentation	51
7.9.2.1	dmac	51
7.9.2.2	smac	51
7.9.2.3	snap	51
7.10	vtss_ace_t Struct Reference	51
7.10.1	Detailed Description	52
7.10.2	Field Documentation	52
7.10.2.1	id	52
7.10.2.2	port_no	52

7.10.2.3	policy	53
7.10.2.4	type	53
7.10.2.5	action	53
7.10.2.6	dmac_mc	53
7.10.2.7	dmac_bc	53
7.10.2.8	vlan	54
7.10.2.9	etype	54
7.10.2.10	llc	54
7.10.2.11	snap	54
7.10.2.12	arp	54
7.10.2.13	ipv4	55
7.10.2.14	ipv6	55
7.10.2.15	frame	55
7.11	vtss_ace_vlan_t Struct Reference	55
7.11.1	Detailed Description	55
7.11.2	Field Documentation	56
7.11.2.1	vid	56
7.11.2.2	usr_prio	56
7.11.2.3	cfi	56
7.12	vtss_acl_action_t Struct Reference	56
7.12.1	Detailed Description	57
7.12.2	Field Documentation	57
7.12.2.1	cpu	57
7.12.2.2	cpu_once	57
7.12.2.3	cpu_queue	57
7.12.2.4	police	58
7.12.2.5	policer_no	58
7.12.2.6	learn	58
7.12.2.7	forward	58
7.12.2.8	port_forward	58

7.12.2.9 port_no	59
7.12.2.10 irq_trigger	59
7.13 vtss_acl_policer_conf_t Struct Reference	59
7.13.1 Detailed Description	59
7.13.2 Field Documentation	59
7.13.2.1 rate	60
7.14 vtss_acl_port_conf_t Struct Reference	60
7.14.1 Detailed Description	60
7.14.2 Field Documentation	60
7.14.2.1 policy_no	60
7.14.2.2 action	61
7.15 vtss_aggr_mode_t Struct Reference	61
7.15.1 Detailed Description	61
7.15.2 Field Documentation	61
7.15.2.1 smac_enable	61
7.15.2.2 dmac_enable	62
7.15.2.3 sip_dip_enable	62
7.15.2.4 sport_dport_enable	62
7.16 vtss_aneg_t Struct Reference	62
7.16.1 Detailed Description	62
7.16.2 Field Documentation	63
7.16.2.1 obey_pause	63
7.16.2.2 generate_pause	63
7.17 vtss_api_lock_t Struct Reference	63
7.17.1 Detailed Description	63
7.17.2 Field Documentation	64
7.17.2.1 inst	64
7.17.2.2 function	64
7.17.2.3 file	64
7.17.2.4 line	64

7.18 vtss_basic_counters_t Struct Reference	65
7.18.1 Detailed Description	65
7.18.2 Field Documentation	65
7.18.2.1 rx_frames	65
7.18.2.2 tx_frames	65
7.19 vtss_chip_id_t Struct Reference	65
7.19.1 Detailed Description	66
7.19.2 Field Documentation	66
7.19.2.1 part_number	66
7.19.2.2 revision	66
7.20 vtss_counter_pair_t Struct Reference	66
7.20.1 Detailed Description	67
7.20.2 Field Documentation	67
7.20.2.1 frames	67
7.20.2.2 bytes	67
7.21 vtss_debug_info_t Struct Reference	67
7.21.1 Detailed Description	68
7.21.2 Field Documentation	68
7.21.2.1 layer	68
7.21.2.2 group	68
7.21.2.3 chip_no	68
7.21.2.4 port_list	68
7.21.2.5 full	69
7.21.2.6 clear	69
7.21.2.7 vml_format	69
7.22 vtss_debug_lock_t Struct Reference	69
7.22.1 Detailed Description	69
7.22.2 Field Documentation	70
7.22.2.1 chip_no	70
7.23 vtss_dgroup_port_conf_t Struct Reference	70

7.23.1 Detailed Description	70
7.23.2 Field Documentation	70
7.23.2.1 dgroup_no	70
7.24 vtss_dlb_policer_conf_t Struct Reference	71
7.24.1 Detailed Description	71
7.24.2 Field Documentation	71
7.24.2.1 type	71
7.24.2.2 enable	71
7.24.2.3 cm	72
7.24.2.4 cf	72
7.24.2.5 line_rate	72
7.24.2.6 cir	72
7.24.2.7 cbs	72
7.24.2.8 eir	73
7.24.2.9 ebs	73
7.25 vtss_ece_action_t Struct Reference	73
7.25.1 Detailed Description	73
7.25.2 Field Documentation	73
7.25.2.1 dir	74
7.25.2.2 pop_tag	74
7.25.2.3 outer_tag	74
7.25.2.4 inner_tag	74
7.25.2.5 policer_id	74
7.25.2.6 evc_id	75
7.25.2.7 policy_no	75
7.26 vtss_ece_frame_ipv4_t Struct Reference	75
7.26.1 Detailed Description	75
7.26.2 Field Documentation	75
7.26.2.1 dscp	76
7.27 vtss_ece_frame_ipv6_t Struct Reference	76

7.27.1	Detailed Description	76
7.27.2	Field Documentation	76
7.27.2.1	dscp	76
7.28	vtss_ece_inner_tag_t Struct Reference	77
7.28.1	Detailed Description	77
7.28.2	Field Documentation	77
7.28.2.1	type	77
7.28.2.2	vid	77
7.28.2.3	pcp_dei_preserve	78
7.28.2.4	pcp	78
7.28.2.5	dei	78
7.29	vtss_ece_key_t Struct Reference	78
7.29.1	Detailed Description	79
7.29.2	Field Documentation	79
7.29.2.1	port_list	79
7.29.2.2	mac	79
7.29.2.3	tag	79
7.29.2.4	inner_tag	79
7.29.2.5	type	80
7.29.2.6	ipv4	80
7.29.2.7	ipv6	80
7.29.2.8	frame	80
7.30	vtss_ece_mac_t Struct Reference	80
7.30.1	Detailed Description	81
7.30.2	Field Documentation	81
7.30.2.1	dmac	81
7.31	vtss_ece_outer_tag_t Struct Reference	81
7.31.1	Detailed Description	81
7.31.2	Field Documentation	82
7.31.2.1	enable	82

7.31.2.2	vid	82
7.31.2.3	pcp_dei_preserve	82
7.31.2.4	pcp	82
7.31.2.5	dei	83
7.32	vtss_ece_t Struct Reference	83
7.32.1	Detailed Description	83
7.32.2	Field Documentation	83
7.32.2.1	id	83
7.32.2.2	key	84
7.32.2.3	action	84
7.33	vtss_ece_tag_t Struct Reference	84
7.33.1	Detailed Description	84
7.33.2	Field Documentation	84
7.33.2.1	vid	85
7.33.2.2	pcp	85
7.33.2.3	dei	85
7.33.2.4	tagged	85
7.33.2.5	s_tagged	85
7.34	vtss_eee_port_conf_t Struct Reference	86
7.34.1	Detailed Description	86
7.34.2	Field Documentation	86
7.34.2.1	eee_ena	86
7.34.2.2	eee_fast_queues	86
7.34.2.3	tx_tw	87
7.34.2.4	lp_advertisement	87
7.34.2.5	optimized_for_power	87
7.35	vtss_eee_port_counter_t Struct Reference	87
7.35.1	Detailed Description	88
7.35.2	Field Documentation	88
7.35.2.1	fill_level_get	88

7.35.2.2 <code>fill_level_thres</code>	88
7.35.2.3 <code>fill_level</code>	88
7.35.2.4 <code>tx_out_bytes_get</code>	88
7.35.2.5 <code>tx_out_bytes</code>	89
7.36 <code>vtss_eee_port_state_t</code> Struct Reference	89
7.36.1 Detailed Description	89
7.36.2 Field Documentation	89
7.36.2.1 <code>select</code>	89
7.36.2.2 <code>val</code>	90
7.37 <code>vtss_eps_port_conf_t</code> Struct Reference	90
7.37.1 Detailed Description	90
7.37.2 Field Documentation	90
7.37.2.1 <code>type</code>	90
7.37.2.2 <code>port_no</code>	91
7.38 <code>vtss_evc_conf_t</code> Struct Reference	91
7.38.1 Detailed Description	91
7.38.2 Field Documentation	91
7.38.2.1 <code>policer_id</code>	91
7.38.2.2 <code>learning</code>	92
7.38.2.3 <code>pb</code>	92
7.38.2.4 <code>network</code>	92
7.39 <code>vtss_evc_counters_t</code> Struct Reference	92
7.39.1 Detailed Description	93
7.39.2 Field Documentation	93
7.39.2.1 <code>rx_green</code>	93
7.39.2.2 <code>rx_yellow</code>	93
7.39.2.3 <code>rx_red</code>	93
7.39.2.4 <code>rx_discard</code>	93
7.39.2.5 <code>tx_discard</code>	94
7.39.2.6 <code>tx_green</code>	94

7.39.2.7	tx_yellow	94
7.40	vtss_evc_pb_conf_t Struct Reference	94
7.40.1	Detailed Description	95
7.40.2	Field Documentation	95
7.40.2.1	nni	95
7.40.2.2	ivid	95
7.40.2.3	vid	95
7.40.2.4	leaf_ivid	95
7.40.2.5	leaf_vid	96
7.40.2.6	leaf	96
7.41	vtss_evc_port_conf_t Struct Reference	96
7.41.1	Detailed Description	96
7.41.2	Field Documentation	96
7.41.2.1	dei_coloring	97
7.42	vtss_ewis_aisl_cons_act_s Struct Reference	97
7.42.1	Detailed Description	97
7.42.2	Field Documentation	97
7.42.2.1	ais_on_los	97
7.42.2.2	ais_on_lof	98
7.43	vtss_ewis_conf_s Struct Reference	98
7.43.1	Detailed Description	98
7.43.2	Field Documentation	98
7.43.2.1	ewis_init_done	99
7.43.2.2	static_conf	99
7.43.2.3	ewis_mode	99
7.43.2.4	section_cons_act	99
7.43.2.5	section_txти	99
7.43.2.6	force_mode	100
7.43.2.7	path_txти	100
7.43.2.8	tx_oh	100

7.43.2.9 tx_oh_passthru	100
7.43.2.10 exp_sl	100
7.43.2.11 test_conf	101
7.43.2.12 ewis_cntr_thresh_conf	101
7.43.2.13 perf_mode	101
7.44 vtss_ewis_cons_act_s Struct Reference	101
7.44.1 Detailed Description	101
7.44.2 Field Documentation	102
7.44.2.1 aisI	102
7.44.2.2 rdil	102
7.44.2.3 fault	102
7.45 vtss_ewis_counter_s Struct Reference	102
7.45.1 Detailed Description	103
7.45.2 Field Documentation	103
7.45.2.1 pn_ebc_p	103
7.45.2.2 pf_ebc_p	103
7.45.2.3 pn_ebc_l	103
7.45.2.4 pf_ebc_l	103
7.45.2.5 pn_ebc_s	104
7.46 vtss_ewis_counter_threshold_s Struct Reference	104
7.46.1 Detailed Description	104
7.46.2 Field Documentation	104
7.46.2.1 n_ebc_thr_s	104
7.46.2.2 n_ebc_thr_l	105
7.46.2.3 f_ebc_thr_l	105
7.46.2.4 n_ebc_thr_p	105
7.46.2.5 f_ebc_thr_p	105
7.47 vtss_ewis_defects_s Struct Reference	105
7.47.1 Detailed Description	106
7.47.2 Field Documentation	106

7.47.2.1 <code>dlos_s</code>	106
7.47.2.2 <code>doof_s</code>	106
7.47.2.3 <code>dlof_s</code>	107
7.47.2.4 <code>dais_l</code>	107
7.47.2.5 <code>drdi_l</code>	107
7.47.2.6 <code>dais_p</code>	107
7.47.2.7 <code>dlop_p</code>	107
7.47.2.8 <code>duneq_p</code>	108
7.47.2.9 <code>drdi_p</code>	108
7.47.2.10 <code>dlcd_p</code>	108
7.47.2.11 <code>dplm_p</code>	108
7.47.2.12 <code>dfais_p</code>	108
7.47.2.13 <code>dfplm_p</code>	109
7.47.2.14 <code>dfuneq_p</code>	109
7.48 <code>vtss_ewis_fault_cons_act_s</code> Struct Reference	109
7.48.1 Detailed Description	109
7.48.2 Field Documentation	110
7.48.2.1 <code>fault_on_feplmp</code>	110
7.48.2.2 <code>fault_on_feaisp</code>	110
7.48.2.3 <code>fault_on_rdil</code>	110
7.48.2.4 <code>fault_on_sef</code>	110
7.48.2.5 <code>fault_on_lof</code>	110
7.48.2.6 <code>fault_on_los</code>	111
7.48.2.7 <code>fault_on_aisl</code>	111
7.48.2.8 <code>fault_on_lcdp</code>	111
7.48.2.9 <code>fault_on_plmp</code>	111
7.48.2.10 <code>fault_on aispl</code>	111
7.48.2.11 <code>fault_on_lopp</code>	112
7.49 <code>vtss_ewis_force_mode_s</code> Struct Reference	112
7.49.1 Detailed Description	112

7.49.2 Field Documentation	112
7.49.2.1 line_rx_force	112
7.49.2.2 line_tx_force	113
7.49.2.3 path_force	113
7.50 vtss_ewis_line_force_mode_s Struct Reference	113
7.50.1 Detailed Description	113
7.50.2 Field Documentation	113
7.50.2.1 force_ais	114
7.50.2.2 force_rdi	114
7.51 vtss_ewis_line_tx_force_mode_s Struct Reference	114
7.51.1 Detailed Description	114
7.51.2 Field Documentation	114
7.51.2.1 force_ais	115
7.51.2.2 force_rdi	115
7.52 vtss_ewis_path_force_mode_s Struct Reference	115
7.52.1 Detailed Description	115
7.52.2 Field Documentation	115
7.52.2.1 force_uneq	116
7.52.2.2 force_rdi	116
7.53 vtss_ewis_perf_mode_s Struct Reference	116
7.53.1 Detailed Description	116
7.53.2 Field Documentation	116
7.53.2.1 pn_ebc_mode_s	117
7.53.2.2 pn_ebc_mode_l	117
7.53.2.3 pf_ebc_mode_l	117
7.53.2.4 pn_ebc_mode_p	117
7.53.2.5 pf_ebc_mode_p	117
7.54 vtss_ewis_perf_s Struct Reference	118
7.54.1 Detailed Description	118
7.54.2 Field Documentation	118

7.54.2.1	pn_ebc_s	118
7.54.2.2	pn_ebc_l	118
7.54.2.3	pf_ebc_l	119
7.54.2.4	pn_ebc_p	119
7.54.2.5	pf_ebc_p	119
7.55	vtss_ewis_rdil_cons_act_s Struct Reference	119
7.55.1	Detailed Description	120
7.55.2	Field Documentation	120
7.55.2.1	rdil_on_los	120
7.55.2.2	rdil_on_lof	120
7.55.2.3	rdil_on_lopc	120
7.55.2.4	rdil_on_ais_l	120
7.56	vtss_ewis_sl_conf_s Struct Reference	121
7.56.1	Detailed Description	121
7.56.2	Field Documentation	121
7.56.2.1	exsl	121
7.57	vtss_ewis_static_conf_s Struct Reference	121
7.57.1	Detailed Description	122
7.57.2	Field Documentation	122
7.57.2.1	ewis_txctrl1	122
7.57.2.2	ewis_txctrl2	122
7.57.2.3	ewis_rx_ctrl1	123
7.57.2.4	ewis_mode_ctrl	123
7.57.2.5	ewis_tx_a1_a2	123
7.57.2.6	ewis_tx_c2_h1	123
7.57.2.7	ewis_tx_h2_h3	123
7.57.2.8	ewis_tx_z0_e1	124
7.57.2.9	ewis_rx_frm_ctrl1	124
7.57.2.10	ewis_rx_frm_ctrl2	124
7.57.2.11	ewis_lof_ctrl1	124

7.57.2.12 ewis_lof_ctrl2	124
7.57.2.13 ewis_rx_err_frc1	125
7.57.2.14 ewis_pmtick_ctrl	125
7.57.2.15 ewis_cnt_cfg	125
7.58 vtss_ewis_status_s Struct Reference	125
7.58.1 Detailed Description	125
7.58.2 Field Documentation	126
7.58.2.1 fault	126
7.58.2.2 link_stat	126
7.59 vtss_ewis_test_conf_s Struct Reference	126
7.59.1 Detailed Description	126
7.59.2 Field Documentation	127
7.59.2.1 loopback	127
7.59.2.2 test_pattern_gen	127
7.59.2.3 test_pattern_ana	127
7.60 vtss_ewis_test_status_s Struct Reference	127
7.60.1 Detailed Description	128
7.60.2 Field Documentation	128
7.60.2.1 tstpat_cnt	128
7.60.2.2 ana_sync	128
7.61 vtss_ewis_tti_s Struct Reference	128
7.61.1 Detailed Description	129
7.61.2 Field Documentation	129
7.61.2.1 mode	129
7.61.2.2 tti	129
7.61.2.3 valid	129
7.62 vtss_ewis_tx_oh_s Struct Reference	129
7.62.1 Detailed Description	130
7.62.2 Field Documentation	130
7.62.2.1 tx_dcc_s	130

7.62.2.2 tx_e1	130
7.62.2.3 tx_f1	131
7.62.2.4 tx_z0	131
7.62.2.5 tx_dcc_l	131
7.62.2.6 tx_e2	131
7.62.2.7 tx_k1_k2	131
7.62.2.8 tx_s1	132
7.62.2.9 tx_z1_z2	132
7.62.2.10 tx_c2	132
7.62.2.11 tx_f2	132
7.62.2.12 tx_n1	132
7.62.2.13 tx_z3_z4	133
7.63 vtss_ewis_tx_passthru_s Struct Reference	133
7.63.1 Detailed Description	133
7.63.2 Field Documentation	133
7.63.2.1 tx_j0	134
7.63.2.2 tx_z0	134
7.63.2.3 tx_b1	134
7.63.2.4 tx_e1	134
7.63.2.5 tx_f1	134
7.63.2.6 tx_dcc_s	135
7.63.2.7 tx_soh	135
7.63.2.8 tx_b2	135
7.63.2.9 tx_k1	135
7.63.2.10 tx_k2	135
7.63.2.11 tx_reil	136
7.63.2.12 tx_dcc_l	136
7.63.2.13 tx_s1	136
7.63.2.14 tx_e2	136
7.63.2.15 tx_z1_z2	136

7.63.2.16 <code>tx_loh</code>	137
7.64 <code>vtss_fan_conf_t</code> Struct Reference	137
7.64.1 Detailed Description	137
7.64.2 Field Documentation	137
7.64.2.1 <code>fan_pwm_freq</code>	137
7.64.2.2 <code>fan_low_pol</code>	138
7.64.2.3 <code>fan_open_col</code>	138
7.64.2.4 <code>type</code>	138
7.64.2.5 <code>ppr</code>	138
7.65 <code>vtss_gpio_10g_gpio_mode_t</code> Struct Reference	138
7.65.1 Detailed Description	139
7.65.2 Field Documentation	139
7.65.2.1 <code>mode</code>	139
7.65.2.2 <code>port</code>	139
7.65.2.3 <code>input</code>	140
7.65.2.4 <code>in_sig</code>	140
7.65.2.5 <code>p_gpio</code>	140
7.65.2.6 <code>c_intrpt</code>	140
7.65.2.7 <code>source</code>	140
7.65.2.8 <code>aggr_intrpt</code>	141
7.65.2.9 <code>use_as_intrpt</code>	141
7.65.2.10 <code>p_gpio_intrpt</code>	141
7.66 <code>vtss_init_conf_t</code> Struct Reference	141
7.66.1 Detailed Description	142
7.66.2 Field Documentation	142
7.66.2.1 <code>reg_read</code>	142
7.66.2.2 <code>reg_write</code>	142
7.66.2.3 <code>miim_read</code>	142
7.66.2.4 <code>miim_write</code>	142
7.66.2.5 <code>mmd_read</code>	143

7.66.2.6 mmd_read_inc	143
7.66.2.7 mmd_write	143
7.66.2.8 spi_read_write	143
7.66.2.9 spi_32bit_read_write	143
7.66.2.10 spi_64bit_read_write	144
7.66.2.11 warm_start_enable	144
7.66.2.12 restart_info_src	144
7.66.2.13 restart_info_port	144
7.66.2.14 mux_mode	144
7.66.2.15 pi	145
7.66.2.16 serdes	145
7.66.2.17 qs_conf	145
7.67 vtss_inst_create_t Struct Reference	145
7.67.1 Detailed Description	145
7.67.2 Field Documentation	146
7.67.2.1 target	146
7.68 vtss_ip_addr_t Struct Reference	146
7.68.1 Detailed Description	146
7.68.2 Field Documentation	146
7.68.2.1 type	146
7.68.2.2 ipv4	147
7.68.2.3 ipv6	147
7.68.2.4 addr	147
7.69 vtss_ip_network_t Struct Reference	147
7.69.1 Detailed Description	147
7.69.2 Field Documentation	148
7.69.2.1 address	148
7.69.2.2 prefix_size	148
7.70 vtss_ipv4_network_t Struct Reference	148
7.70.1 Detailed Description	148

7.70.2 Field Documentation	148
7.70.2.1 address	149
7.70.2.2 prefix_size	149
7.71 vtss_ipv4_uc_t Struct Reference	149
7.71.1 Detailed Description	149
7.71.2 Field Documentation	149
7.71.2.1 network	150
7.71.2.2 destination	150
7.72 vtss_ipv6_network_t Struct Reference	150
7.72.1 Detailed Description	150
7.72.2 Field Documentation	150
7.72.2.1 address	151
7.72.2.2 prefix_size	151
7.73 vtss_ipv6_t Struct Reference	151
7.73.1 Detailed Description	151
7.73.2 Field Documentation	151
7.73.2.1 addr	152
7.74 vtss_ipv6_uc_t Struct Reference	152
7.74.1 Detailed Description	152
7.74.2 Field Documentation	152
7.74.2.1 network	152
7.74.2.2 destination	153
7.75 vtss_irq_conf_t Struct Reference	153
7.75.1 Detailed Description	153
7.75.2 Field Documentation	153
7.75.2.1 external	153
7.75.2.2 destination	154
7.76 vtss_irq_status_t Struct Reference	154
7.76.1 Detailed Description	154
7.76.2 Field Documentation	154

7.76.2.1	active	154
7.76.2.2	raw_ident	155
7.76.2.3	raw_status	155
7.76.2.4	raw_mask	155
7.77	vtss_l3_common_conf_t Struct Reference	155
7.77.1	Detailed Description	155
7.77.2	Field Documentation	156
7.77.2.1	rleg_mode	156
7.77.2.2	base_address	156
7.77.2.3	routing_enable	156
7.78	vtss_l3_counters_t Struct Reference	156
7.78.1	Detailed Description	157
7.78.2	Field Documentation	157
7.78.2.1	ipv4uc_received_octets	157
7.78.2.2	ipv4uc_received_frames	157
7.78.2.3	ipv6uc_received_octets	157
7.78.2.4	ipv6uc_received_frames	157
7.78.2.5	ipv4uc_transmitted_octets	158
7.78.2.6	ipv4uc_transmitted_frames	158
7.78.2.7	ipv6uc_transmitted_octets	158
7.78.2.8	ipv6uc_transmitted_frames	158
7.79	vtss_l3_neighbour_t Struct Reference	158
7.79.1	Detailed Description	159
7.79.2	Field Documentation	159
7.79.2.1	dmac	159
7.79.2.2	vlan	159
7.79.2.3	dip	159
7.80	vtss_l3_rleg_conf_t Struct Reference	160
7.80.1	Detailed Description	160
7.80.2	Field Documentation	160

7.80.2.1	ipv4_unicast_enable	160
7.80.2.2	ipv6_unicast_enable	160
7.80.2.3	ipv4_icmp_redirect_enable	161
7.80.2.4	ipv6_icmp_redirect_enable	161
7.80.2.5	vlan	161
7.80.2.6	vrid0_enable	161
7.80.2.7	vrid0	162
7.80.2.8	vrid1_enable	162
7.80.2.9	vrid1	162
7.81	vtss_lcpll_status_t Struct Reference	162
7.81.1	Detailed Description	163
7.81.2	Field Documentation	163
7.81.2.1	lock_status	163
7.81.2.2	cal_done	163
7.81.2.3	cal_error	163
7.81.2.4	fsm_lock	163
7.81.2.5	fsm_stat	164
7.81.2.6	gain_stat	164
7.82	vtss_learn_mode_t Struct Reference	164
7.82.1	Detailed Description	164
7.82.2	Field Documentation	164
7.82.2.1	automatic	165
7.82.2.2	cpu	165
7.82.2.3	discard	165
7.83	vtss_mac_t Struct Reference	165
7.83.1	Detailed Description	165
7.83.2	Field Documentation	166
7.83.2.1	addr	166
7.84	vtss_mac_table_entry_t Struct Reference	166
7.84.1	Detailed Description	166

7.84.2 Field Documentation	166
7.84.2.1 vid_mac	167
7.84.2.2 destination	167
7.84.2.3 copy_to_cpu	167
7.84.2.4 locked	167
7.84.2.5 aged	167
7.84.2.6 cpu_queue	168
7.84.2.7 enable	168
7.84.2.8 remote_entry	168
7.84.2.9 upsid	168
7.84.2.10 upspn	168
7.84.2.11 vstax2	169
7.85 vtss_mac_table_status_t Struct Reference	169
7.85.1 Detailed Description	169
7.85.2 Field Documentation	169
7.85.2.1 learned	169
7.85.2.2 replaced	170
7.85.2.3 moved	170
7.85.2.4 aged	170
7.86 vtss_mirror_conf_t Struct Reference	170
7.86.1 Detailed Description	171
7.86.2 Field Documentation	171
7.86.2.1 port_no	171
7.86.2.2 fwd_enable	171
7.86.2.3 tag	171
7.86.2.4 vid	171
7.86.2.5 pcp	172
7.86.2.6 dei	172
7.87 vtss_mtimer_t Struct Reference	172
7.87.1 Detailed Description	172

7.87.2 Field Documentation	172
7.87.2.1 timeout	173
7.87.2.2 now	173
7.88 vtss_npi_conf_t Struct Reference	173
7.88.1 Detailed Description	173
7.88.2 Field Documentation	173
7.88.2.1 enable	174
7.88.2.2 port_no	174
7.89 vtss_os_timestamp_t Struct Reference	174
7.89.1 Detailed Description	174
7.89.2 Field Documentation	174
7.89.2.1 hw_cnt	175
7.90 vtss_packet_dma_conf_t Struct Reference	175
7.90.1 Detailed Description	175
7.90.2 Field Documentation	175
7.90.2.1 dma_enable	175
7.91 vtss_packet_frame_info_t Struct Reference	176
7.91.1 Detailed Description	176
7.91.2 Field Documentation	176
7.91.2.1 port_no	176
7.91.2.2 glag_no	176
7.91.2.3 vid	177
7.91.2.4 port_tx	177
7.91.2.5 aggr_rx_disable	177
7.91.2.6 aggr_tx_disable	177
7.92 vtss_packet_port_filter_t Struct Reference	177
7.92.1 Detailed Description	178
7.92.2 Field Documentation	178
7.92.2.1 filter	178
7.92.2.2 tpid	178

7.93 vtss_packet_port_info_t Struct Reference	178
7.93.1 Detailed Description	179
7.93.2 Field Documentation	179
7.93.2.1 port_no	179
7.93.2.2 glag_no	179
7.93.2.3 vid	179
7.93.2.4 aggr_rx_disable	180
7.93.2.5 aggr_tx_disable	180
7.94 vtss_packet_rx_conf_t Struct Reference	180
7.94.1 Detailed Description	180
7.94.2 Field Documentation	180
7.94.2.1 queue	181
7.94.2.2 reg	181
7.94.2.3 map	181
7.94.2.4 grp_map	181
7.95 vtss_packet_rx_header_t Struct Reference	181
7.95.1 Detailed Description	182
7.95.2 Field Documentation	182
7.95.2.1 length	182
7.95.2.2 port_no	182
7.95.2.3 queue_mask	182
7.95.2.4 learn	183
7.95.2.5 arrived_tagged	183
7.95.2.6 tag	183
7.95.2.7 vstax	183
7.96 vtss_packet_rx_info_t Struct Reference	183
7.96.1 Detailed Description	184
7.96.2 Field Documentation	184
7.96.2.1 hints	185
7.96.2.2 length	185

7.96.2.3 port_no	185
7.96.2.4 glag_no	186
7.96.2.5 tag_type	186
7.96.2.6 tag	187
7.96.2.7 stripped_tag	187
7.96.2.8 xtr_qu_mask	188
7.96.2.9 cos	188
7.96.2.10 acl_hit	188
7.96.2.11 acl_idx	189
7.96.2.12 sw_tstamp	189
7.96.2.13 tstamp_id	189
7.96.2.14 tstamp_id_decoded	190
7.96.2.15 hw_tstamp	190
7.96.2.16 hw_tstamp_decoded	190
7.96.2.17 sflow_type	191
7.96.2.18 sflow_port_no	191
7.96.2.19 oam_info	192
7.96.2.20 oam_info_decoded	192
7.96.2.21 isdx	192
7.96.2.22 vstax	193
7.97 vtss_packet_rx_meta_t Struct Reference	193
7.97.1 Detailed Description	193
7.97.2 Field Documentation	194
7.97.2.1 no_wait	194
7.97.2.2 chip_no	194
7.97.2.3 xtr_qu	195
7.97.2.4 etype	195
7.97.2.5 fcs	196
7.97.2.6 sw_tstamp	196
7.97.2.7 length	197

7.98 vtss_packet_rx_port_conf_t Struct Reference	197
7.98.1 Detailed Description	197
7.98.2 Field Documentation	198
7.98.2.1 bpdu_reg	198
7.98.2.2 garp_reg	198
7.99 vtss_packet_rx_queue_conf_t Struct Reference	198
7.99.1 Detailed Description	198
7.99.2 Field Documentation	198
7.99.2.1 size	199
7.99.2.2 npi	199
7.100vtss_packet_rx_queue_map_t Struct Reference	199
7.100.1 Detailed Description	199
7.100.2 Field Documentation	200
7.100.2.1 bpdu_queue	200
7.100.2.2 garp_queue	200
7.100.2.3 learn_queue	200
7.100.2.4 igmp_queue	200
7.100.2.5 ipmc_ctrl_queue	200
7.100.2.6 mac_vid_queue	201
7.100.2.7 stack_queue	201
7.100.2.8 sflow_queue	201
7.100.2.9 lrn_all_queue	201
7.100.2.103_uc_queue	201
7.100.2.113_other_queue	202
7.101vtss_packet_rx_queue_npi_conf_t Struct Reference	202
7.101.1 Detailed Description	202
7.101.2 Field Documentation	202
7.101.2.1 enable	202
7.102vtss_packet_rx_reg_t Struct Reference	203
7.102.1 Detailed Description	203

7.102.2 Field Documentation	203
7.102.2.1 bpdu_cpu_only	203
7.102.2.2 garp_cpu_only	203
7.102.2.3 ipmc_ctrl_cpu_copy	204
7.102.2.4 igmp_cpu_only	204
7.102.2.5 mld_cpu_only	204
7.103vtss_packet_tx_ifh_t Struct Reference	204
7.103.1 Detailed Description	204
7.103.2 Field Documentation	205
7.103.2.1 length	205
7.103.2.2 ifh	205
7.104vtss_packet_tx_info_t Struct Reference	205
7.104.1 Detailed Description	206
7.104.2 Field Documentation	206
7.104.2.1 switch_frm	206
7.104.2.2 dst_port_mask	207
7.104.2.3 frm_len	207
7.104.2.4 tag	208
7.104.2.5 aggr_code	208
7.104.2.6 cos	209
7.104.2.7 tx_vstax_hdr	210
7.104.2.8 bin	210
7.104.2.9 sym	211
7.104.2.10vstax	211
7.104.2.11ptp_action	211
7.104.2.12ptp_id	212
7.104.2.13ptp_timestamp	212
7.104.2.14atch_timestamp	213
7.104.2.15bam_type	213
7.104.2.16sdx	214

7.104.2.17sdx_dont_use	214
7.104.2.18dp	215
7.104.2.19masquerade_port	215
7.104.2.20pdu_offset	216
7.105vtss_phy_10g_apc_conf_t Struct Reference	216
7.105.1 Detailed Description	216
7.105.2 Field Documentation	216
7.105.2.1 op_mode	217
7.105.2.2 op_mode_flag	217
7.106vtss_phy_10g_apc_status_t Struct Reference	217
7.106.1 Detailed Description	217
7.106.2 Field Documentation	217
7.106.2.1 reset	218
7.106.2.2 freeze	218
7.107vtss_phy_10g_auto_failover_conf_t Struct Reference	218
7.107.1 Detailed Description	218
7.107.2 Field Documentation	219
7.107.2.1 port_no	219
7.107.2.2 evnt	219
7.107.2.3 trig_ch_id	219
7.107.2.4 is_host_side	219
7.107.2.5 channel_id	219
7.107.2.6 v_gpio	220
7.107.2.7 a_gpio	220
7.107.2.8 enable	220
7.107.2.9 filter	220
7.107.2.10ltr_val	220
7.108vtss_phy_10g_base_kr_autoneg_t Struct Reference	221
7.108.1 Detailed Description	221
7.108.2 Field Documentation	221

7.108.2.1 an_restart	221
7.108.2.2 an_enable	221
7.108.2.3 an_reset	222
7.109vtss_phy_10g_base_kr_conf_t Struct Reference	222
7.109.1 Detailed Description	222
7.109.2 Field Documentation	222
7.109.2.1 cm1	222
7.109.2.2 c0	223
7.109.2.3 c1	223
7.109.2.4 ampl	223
7.109.2.5 slewrate	223
7.109.2.6 en_ob	223
7.109.2.7 ser_inv	224
7.110vtss_phy_10g_base_kr_id_adv_abil_t Struct Reference	224
7.110.1 Detailed Description	224
7.110.2 Field Documentation	224
7.110.2.1 adv_1g	224
7.110.2.2 adv_10g	225
7.110.2.3 fec_abil	225
7.110.2.4 fec_req	225
7.111vtss_phy_10g_base_kr_status_t Struct Reference	225
7.111.1 Detailed Description	225
7.111.2 Field Documentation	226
7.111.2.1 aneg	226
7.111.2.2 train	226
7.111.2.3 fec	226
7.112vtss_phy_10g_base_kr_train_aneg_t Struct Reference	226
7.112.1 Detailed Description	227
7.112.2 Field Documentation	227
7.112.2.1 training	227

7.112.2.2 autoneg	227
7.112.2.3 ld_abil	227
7.112.2.4 host_kr	227
7.112.2.5 line_kr	228
7.113vtss_phy_10g_base_kr_training_t Struct Reference	228
7.113.1 Detailed Description	228
7.113.2 Field Documentation	228
7.113.2.1 enable	228
7.113.2.2 trmthd_cp	229
7.113.2.3 trmthd_c0	229
7.113.2.4 trmthd_cm	229
7.113.2.5 ld_pre_init	229
7.114vtss_phy_10g_ckout_conf_t Struct Reference	229
7.114.1 Detailed Description	230
7.114.2 Field Documentation	230
7.114.2.1 mode	230
7.114.2.2 src	230
7.114.2.3 freq	230
7.114.2.4 squelch_inv	231
7.114.2.5 enable	231
7.114.2.6 ckout_sel	231
7.115vtss_phy_10g_clause_37_adv_t Struct Reference	231
7.115.1 Detailed Description	232
7.115.2 Field Documentation	232
7.115.2.1 fdx	232
7.115.2.2 hdx	232
7.115.2.3 symmetric_pause	232
7.115.2.4 asymmetric_pause	232
7.115.2.5 remote_fault	233
7.115.2.6 acknowledge	233

7.115.2.7 next_page	233
7.116vtss_phy_10g_clause_37_cmn_status_t Struct Reference	233
7.116.1 Detailed Description	233
7.116.2 Field Documentation	234
7.116.2.1 line	234
7.116.2.2 host	234
7.117vtss_phy_10g_clause_37_control_t Struct Reference	234
7.117.1 Detailed Description	234
7.117.2 Field Documentation	235
7.117.2.1 enable	235
7.117.2.2 advertisement	235
7.117.2.3 enable_pass_thru	235
7.117.2.4 line	235
7.117.2.5 host	235
7.117.2.6 l_h	236
7.118vtss_phy_10g_clause_37_status_t Struct Reference	236
7.118.1 Detailed Description	236
7.118.2 Field Documentation	236
7.118.2.1 link	236
7.118.2.2 complete	237
7.118.2.3 partner_advertisement	237
7.118.2.4 autoneg	237
7.119vtss_phy_10g_clk_src_t Struct Reference	237
7.119.1 Detailed Description	237
7.119.2 Field Documentation	238
7.119.2.1 is_high_amp	238
7.120vtss_phy_10g_cnt_t Struct Reference	238
7.120.1 Detailed Description	238
7.120.2 Field Documentation	238
7.120.2.1 pcs	238

7.121vtss_phy_10g_fifo_sync_t Struct Reference	239
7.121.1 Detailed Description	239
7.121.2 Field Documentation	239
7.121.2.1 bypass_in_api	239
7.121.2.2 skip_rev_check	239
7.121.2.3 pr	240
7.122vtss_phy_10g_fw_status_t Struct Reference	240
7.122.1 Detailed Description	240
7.122.2 Field Documentation	240
7.122.2.1 edc_fw_rev	240
7.122.2.2 edc_fw_chksum	241
7.122.2.3 icpu_activity	241
7.122.2.4 edc_fw_api_load	241
7.123vtss_phy_10g_host_clk_conf_t Struct Reference	241
7.123.1 Detailed Description	242
7.123.2 Field Documentation	242
7.123.2.1 mode	242
7.123.2.2 recvrd_clk_sel	242
7.123.2.3 clk_sel_no	242
7.124vtss_phy_10g_ib_conf_t Struct Reference	242
7.124.1 Detailed Description	243
7.124.2 Field Documentation	243
7.124.2.1 offs	243
7.124.2.2 gain	243
7.124.2.3 gainadj	244
7.124.2.4 l	244
7.124.2.5 c	244
7.124.2.6 agc	244
7.124.2.7 dfe1	244
7.124.2.8 dfe2	245

7.124.2.9 dfe3	245
7.124.2.10 dfe4	245
7.124.2.11 ld	245
7.124.2.12 prbs	245
7.124.2.13 prbs_inv	246
7.124.2.14 apc_bit_mask	246
7.124.2.15 freeze_bit_mask	246
7.124.2.16 config_bit_mask	246
7.124.2.17 s_host	246
7.125 vtss_phy_10g_ib_status_t Struct Reference	247
7.125.1 Detailed Description	247
7.125.2 Field Documentation	247
7.125.2.1 ib_conf	247
7.125.2.2 sig_det	247
7.125.2.3 bit_errors	248
7.126 vtss_phy_10g_ib_storage_t Struct Reference	248
7.126.1 Detailed Description	248
7.126.2 Field Documentation	248
7.126.2.1 ib_storage_bool	248
7.126.2.2 ib_storage	249
7.127 vtss_phy_10g_id_t Struct Reference	249
7.127.1 Detailed Description	249
7.127.2 Field Documentation	249
7.127.2.1 part_number	249
7.127.2.2 revision	250
7.127.2.3 channel_id	250
7.127.2.4 family	250
7.127.2.5 type	250
7.127.2.6 phy_api_base_no	250
7.127.2.7 device_feature_status	251

7.128vtss_phy_10g_init_parm_t Struct Reference	251
7.128.1 Detailed Description	251
7.128.2 Field Documentation	251
7.128.2.1 channel_conf	251
7.129vtss_phy_10g_jitter_conf_t Struct Reference	252
7.129.1 Detailed Description	252
7.129.2 Field Documentation	252
7.129.2.1 incr_levn	252
7.129.2.2 levn	252
7.129.2.3 vtail	253
7.130vtss_phy_10g_kr_status_aneg_t Struct Reference	253
7.130.1 Detailed Description	253
7.130.2 Field Documentation	253
7.130.2.1 complete	253
7.130.2.2 active	254
7.130.2.3 request_10g	254
7.130.2.4 request_1g	254
7.130.2.5 request_fec_change	254
7.130.2.6 fec_enable	254
7.130.2.7 sm	255
7.130.2.8 lp_aneg_able	255
7.130.2.9 block_lock	255
7.131vtss_phy_10g_kr_status_fec_t Struct Reference	255
7.131.1 Detailed Description	255
7.131.2 Field Documentation	256
7.131.2.1 enable	256
7.131.2.2 corrected_block_cnt	256
7.131.2.3 uncorrected_block_cnt	256
7.132vtss_phy_10g_kr_status_train_t Struct Reference	256
7.132.1 Detailed Description	257

7.132.2 Field Documentation	257
7.132.2.1 complete	257
7.132.2.2 cm_ob_tap_result	257
7.132.2.3 cp_ob_tap_result	257
7.132.2.4 c0_ob_tap_result	257
7.133vtss_phy_10g_lane_sync_conf_t Struct Reference	258
7.133.1 Detailed Description	258
7.133.2 Field Documentation	258
7.133.2.1 enable	258
7.133.2.2 tx_macro	258
7.133.2.3 rx_macro	259
7.133.2.4 rx_ch	259
7.133.2.5 tx_ch	259
7.134vtss_phy_10g_line_clk_conf_t Struct Reference	259
7.134.1 Detailed Description	260
7.134.2 Field Documentation	260
7.134.2.1 mode	260
7.134.2.2 recvrd_clk_sel	260
7.134.2.3 clk_sel_no	260
7.135vtss_phy_10g_loopback_t Struct Reference	260
7.135.1 Detailed Description	261
7.135.2 Field Documentation	261
7.135.2.1 lb_type	261
7.135.2.2 enable	261
7.136vtss_phy_10g_mode_t Struct Reference	261
7.136.1 Detailed Description	263
7.136.2 Member Enumeration Documentation	263
7.136.2.1 anonymous enum	263
7.136.3 Field Documentation	263
7.136.3.1 oper_mode	263

7.136.3.2 interface	263
7.136.3.3 wrefclk	264
7.136.3.4 high_input_gain	264
7.136.3.5 xfi_pol_invert	264
7.136.3.6 xaui_lane_flip	264
7.136.3.7 channel_id	264
7.136.3.8 hl_clk_synth	265
7.136.3.9 rcvrd_clk	265
7.136.3.10rcvrd_clk_div	265
7.136.3.11sref_clk_div	265
7.136.3.12wref_clk_div	265
7.136.3.13edc_fw_load	266
7.136.3.14use_conf	266
7.136.3.15ob_conf	266
7.136.3.16b_conf	266
7.136.3.17dig_offset_reg	266
7.136.3.18apc_offs_ctrl	267
7.136.3.19apc_line_id_ctrl	267
7.136.3.20apc_host_id_ctrl	267
7.136.3.21d_filter	267
7.136.3.22cfg0	267
7.136.3.23b_ini_lp	268
7.136.3.24b_min_lp	268
7.136.3.25b_max_lp	268
7.136.3.26apc_eqz_offs_par_cfg	268
7.136.3.27apc_line_eqz_id_ctrl	268
7.136.3.28apc_host_eqz_id_ctrl	269
7.136.3.29_offset_guard	269
7.136.3.30h_offset_guard	269
7.136.3.31serdes_conf	269

7.136.3.32apc_ib_regulator	269
7.136.3.33pma_txratecontrol	270
7.136.3.34venice_rev_a_los_detection_workaround	270
7.136.3.35ddr_mode	270
7.136.3.36master	270
7.136.3.37rate	270
7.136.3.38polarity	271
7.136.3.39s_host_wan	271
7.136.3.40h_clk_src	271
7.136.3.41l_clk_src	271
7.136.3.42ref_for_host	271
7.136.3.43link_6g_distance	272
7.136.3.44h_media	272
7.136.3.45_media	272
7.136.3.46h_ib_conf	272
7.136.3.47_ib_conf	272
7.136.3.48h_apc_conf	273
7.136.3.49_apc_conf	273
7.136.3.50enable_pass_thru	273
7.136.3.51is_init	273
7.136.3.52sd6g_calib_done	273
7.137vtss_phy_10g_ob_status_t Struct Reference	274
7.137.1 Detailed Description	274
7.137.2 Field Documentation	274
7.137.2.1 r_ctrl	274
7.137.2.2 c_ctrl	274
7.137.2.3 slew	275
7.137.2.4 levn	275
7.137.2.5 d_fltr	275
7.137.2.6 v3	275

7.137.2.7 vp	275
7.137.2.8 v4	276
7.137.2.9 v5	276
7.137.2.10s_host	276
7.138vtss_phy_10g_pcs_prbs_gen_conf_t Struct Reference	276
7.138.1 Detailed Description	276
7.138.2 Field Documentation	277
7.138.2.1 prbs_gen	277
7.139vtss_phy_10g_pcs_prbs_mon_conf_t Struct Reference	277
7.139.1 Detailed Description	277
7.139.2 Field Documentation	277
7.139.2.1 prbs_mon	277
7.139.2.2 error_counter	278
7.140vtss_phy_10g_pkt_gen_conf_t Struct Reference	278
7.140.1 Detailed Description	278
7.140.2 Field Documentation	278
7.140.2.1 enable	279
7.140.2.2 ptp	279
7.140.2.3 ingress	279
7.140.2.4 frames	279
7.140.2.5 frame_single	279
7.140.2.6 etype	280
7.140.2.7 pkt_len	280
7.140.2.8 ipg_len	280
7.140.2.9 smac	280
7.140.2.10dmac	280
7.140.2.11ptp_ts_sec	281
7.140.2.12ptp_ts_ns	281
7.140.2.13srate	281
7.141vtss_phy_10g_pkt_mon_conf_t Struct Reference	281

7.141.1 Detailed Description	282
7.141.2 Field Documentation	282
7.141.2.1 enable	282
7.141.2.2 update	282
7.141.2.3 reset	282
7.141.2.4 good_crc	282
7.141.2.5 bad_crc	283
7.141.2.6 frag	283
7.141.2.7 lfault	283
7.141.2.8 ber	283
7.142vtss_phy_10g_polarity_inv_t Struct Reference	283
7.142.1 Detailed Description	284
7.142.2 Field Documentation	284
7.142.2.1 line_rx	284
7.142.2.2 line_tx	284
7.142.2.3 host_rx	284
7.142.2.4 host_tx	285
7.143vtss_phy_10g_prbs_gen_conf_t Struct Reference	285
7.143.1 Detailed Description	285
7.143.2 Field Documentation	285
7.143.2.1 enable	285
7.143.2.2 prbsn_tx_sel	286
7.143.2.3 line	286
7.143.2.4 prbsn_tx_io	286
7.143.2.5 prbsn_tx_iw	286
7.144vtss_phy_10g_prbs_mon_conf_t Struct Reference	286
7.144.1 Detailed Description	287
7.144.2 Field Documentation	287
7.144.2.1 enable	287
7.144.2.2 line	287

7.144.2.3 max_bist_frames	288
7.144.2.4 error_states	288
7.144.2.5 des_interface_width	288
7.144.2.6 prbsn_sel	288
7.144.2.7 prbs_check_input_invert	288
7.144.2.8 no_of_errors	289
7.144.2.9 bist_mode	289
7.144.2.10error_status	289
7.144.2.11PRBS_status	289
7.144.2.12main_status	289
7.144.2.13stuck_at_par	290
7.144.2.14stuck_at_01	290
7.144.2.15no_sync	290
7.144.2.16nstable	290
7.144.2.17incomplete	290
7.144.2.18active	291
7.145vtss_phy_10g_rxckout_conf_t Struct Reference	291
7.145.1 Detailed Description	291
7.145.2 Field Documentation	291
7.145.2.1 mode	291
7.145.2.2 squelch_on_pcs_fault	292
7.145.2.3 squelch_on_lopc	292
7.146vtss_phy_10g_sckout_conf_t Struct Reference	292
7.146.1 Detailed Description	292
7.146.2 Field Documentation	292
7.146.2.1 mode	293
7.146.2.2 src	293
7.146.2.3 freq	293
7.146.2.4 squelch_inv	293
7.146.2.5 enable	293

7.147vtss_phy_10g_serdes_status_t Struct Reference	294
7.147.1 Detailed Description	294
7.147.2 Field Documentation	294
7.147.2.1 rcomp	295
7.147.2.2 h_pll5g_lock_status	295
7.147.2.3 h_pll5g_fsm_lock	295
7.147.2.4 h_pll5g_fsm_stat	295
7.147.2.5 h_pll5g_gain	295
7.147.2.6 l_pll5g_lock_status	296
7.147.2.7 l_pll5g_fsm_lock	296
7.147.2.8 l_pll5g_fsm_stat	296
7.147.2.9 l_pll5g_gain	296
7.147.2.10h_rx_rcpll_lock_status	296
7.147.2.11h_rx_rcpll_range	297
7.147.2.12h_rx_rcpll_vco_load	297
7.147.2.13h_rx_rcpll_fsm_status	297
7.147.2.14_rx_rcpll_lock_status	297
7.147.2.15_rx_rcpll_range	297
7.147.2.16_rx_rcpll_vco_load	298
7.147.2.17_rx_rcpll_fsm_status	298
7.147.2.18h_tx_rcpll_lock_status	298
7.147.2.19h_tx_rcpll_range	298
7.147.2.20h_tx_rcpll_vco_load	298
7.147.2.21h_tx_rcpll_fsm_status	299
7.147.2.22_tx_rcpll_lock_status	299
7.147.2.23_tx_rcpll_range	299
7.147.2.24_tx_rcpll_vco_load	299
7.147.2.25_tx_rcpll_fsm_status	299
7.147.2.26h_pma	300
7.147.2.27h_pcs	300

7.147.2.28_pma	300
7.147.2.29_pcs	300
7.147.2.30wis	300
7.148vtss_phy_10g_srefclk_mode_t Struct Reference	301
7.148.1 Detailed Description	301
7.148.2 Field Documentation	301
7.148.2.1 enable	301
7.148.2.2 freq	301
7.149vtss_phy_10g_status_t Struct Reference	301
7.149.1 Detailed Description	302
7.149.2 Field Documentation	302
7.149.2.1 pma	302
7.149.2.2 hpma	302
7.149.2.3 wis	302
7.149.2.4 pcs	303
7.149.2.5 hpcs	303
7.149.2.6 xs	303
7.149.2.7 lpcs_1g	303
7.149.2.8 hpcs_1g	303
7.149.2.9 status	304
7.149.2.10block_lock	304
7.149.2.11opc_stat	304
7.150vtss_phy_10g_timestamp_val_t Struct Reference	304
7.150.1 Detailed Description	304
7.150.2 Field Documentation	305
7.150.2.1 timestamp	305
7.151vtss_phy_10g_txckout_conf_t Struct Reference	305
7.151.1 Detailed Description	305
7.151.2 Field Documentation	305
7.151.2.1 mode	305

7.152vtss_phy_10g_vscope_conf_t Struct Reference	306
7.152.1 Detailed Description	306
7.152.2 Field Documentation	306
7.152.2.1 scan_type	306
7.152.2.2 line	306
7.152.2.3 enable	306
7.152.2.4 error_thres	307
7.153vtss_phy_10g_vscope_scan_conf_t Struct Reference	307
7.153.1 Detailed Description	307
7.153.2 Field Documentation	307
7.153.2.1 line	307
7.153.2.2 x_start	308
7.153.2.3 y_start	308
7.153.2.4 x_incr	308
7.153.2.5 y_incr	308
7.153.2.6 x_count	308
7.153.2.7 y_count	309
7.153.2.8 ber	309
7.154vtss_phy_10g_vscope_scan_status_t Struct Reference	309
7.154.1 Detailed Description	309
7.154.2 Field Documentation	309
7.154.2.1 scan_conf	310
7.154.2.2 error_free_x	310
7.154.2.3 error_free_y	310
7.154.2.4 amp_range	310
7.154.2.5 errors	310
7.155vtss_phy_aneg_t Struct Reference	311
7.155.1 Detailed Description	311
7.155.2 Field Documentation	311
7.155.2.1 speed_10m_hdx	311

7.155.2.2 speed_10m_fdx	311
7.155.2.3 speed_100m_hdx	312
7.155.2.4 speed_100m_fdx	312
7.155.2.5 speed_1g_fdx	312
7.155.2.6 speed_1g_hdx	312
7.155.2.7 symmetric_pause	312
7.155.2.8 asymmetric_pause	313
7.155.2.9 tx_remote_fault	313
7.156vtss_phy_clock_conf_t Struct Reference	313
7.156.1 Detailed Description	313
7.156.2 Field Documentation	313
7.156.2.1 src	314
7.156.2.2 freq	314
7.156.2.3 squelch	314
7.157vtss_phy_conf_1g_t Struct Reference	314
7.157.1 Detailed Description	315
7.157.2 Field Documentation	315
7.157.2.1 cfg	315
7.157.2.2 val	315
7.157.2.3 master	315
7.158vtss_phy_conf_t Struct Reference	315
7.158.1 Detailed Description	316
7.158.2 Field Documentation	316
7.158.2.1 mode	316
7.158.2.2 forced	316
7.158.2.3 aneg	316
7.158.2.4 mdi	317
7.158.2.5 flf	317
7.158.2.6 sigdet	317
7.158.2.7 unidir	317

7.158.2.8 mac_if_pcs	317
7.158.2.9 media_if_pcs	318
7.158.2.10 force_ams_sel	318
7.159 vtss_phy_daisy_chain_conf_t Struct Reference	318
7.159.1 Detailed Description	318
7.159.2 Field Documentation	318
7.159.2.1 spi_daisy_input	319
7.159.2.2 spi_daisy_output	319
7.160 vtss_phy_eee_conf_t Struct Reference	319
7.160.1 Detailed Description	319
7.160.2 Field Documentation	319
7.160.2.1 eee_mode	320
7.160.2.2 eee_ena_phy	320
7.161 vtss_phy_enhanced_led_control_t Struct Reference	320
7.161.1 Detailed Description	320
7.161.2 Field Documentation	320
7.161.2.1 ser_led_output_1	321
7.161.2.2 ser_led_output_2	321
7.161.2.3 ser_led_frame_rate	321
7.161.2.4 ser_led_select	321
7.162 vtss_phy_forced_t Struct Reference	321
7.162.1 Detailed Description	322
7.162.2 Field Documentation	322
7.162.2.1 speed	322
7.162.2.2 fdx	322
7.163 vtss_phy_led_mode_select_t Struct Reference	322
7.163.1 Detailed Description	323
7.163.2 Field Documentation	323
7.163.2.1 mode	323
7.163.2.2 number	323

7.164vtss_phy_loopback_t Struct Reference	323
7.164.1 Detailed Description	324
7.164.2 Field Documentation	324
7.164.2.1 far_end_enable	324
7.164.2.2 near_end_enable	324
7.164.2.3 connector_enable	324
7.164.2.4 mac_serdes_input_enable	324
7.164.2.5 mac_serdes_facility_enable	325
7.164.2.6 mac_serdes_equipment_enable	325
7.164.2.7 media_serdes_input_enable	325
7.164.2.8 media_serdes_facility_enable	325
7.164.2.9 media_serdes_equipment_enable	325
7.165vtss_phy_ltc_freq_synth_s Struct Reference	326
7.165.1 Detailed Description	326
7.165.2 Field Documentation	326
7.165.2.1 enable	326
7.165.2.2 high_duty_cycle	326
7.165.2.3 low_duty_cycle	327
7.166vtss_phy_mac_serd_pcs_cntl_t Struct Reference	327
7.166.1 Detailed Description	327
7.166.2 Field Documentation	327
7.166.2.1 disable	328
7.166.2.2 restart	328
7.166.2.3 pd_enable	328
7.166.2.4 aneg_restart	328
7.166.2.5 force_adv_ability	328
7.166.2.6 sgmii_in_pre	329
7.166.2.7 sgmii_out_pre	329
7.166.2.8 serdes_aneg_ena	329
7.166.2.9 serdes_pol_inv_in	329

7.166.2.10 serdes_pol_inv_out	329
7.166.2.11 fast_link_stat_ena	330
7.166.2.12 inhibit_odd_start	330
7.167 vtss_phy_media_serdes_cntl_t Struct Reference	330
7.167.1 Detailed Description	330
7.167.2 Field Documentation	331
7.167.2.1 remote_fault	331
7.167.2.2 aneg_pd_detect	331
7.167.2.3 force_adv_ability	331
7.167.2.4 serdes_pol_inv_in	331
7.167.2.5 serdes_pol_inv_out	331
7.167.2.6 inhibit_odd_start	332
7.167.2.7 force_hls	332
7.167.2.8 force_fefi	332
7.167.2.9 force_fefi_value	332
7.168 vtss_phy_pcs_cntl_t Struct Reference	332
7.168.1 Detailed Description	333
7.168.2 Field Documentation	333
7.168.2.1 block_lock_latched	333
7.168.2.2 high_ber_latched	333
7.168.2.3 ber_cnt	333
7.168.2.4 err_blk_cnt	334
7.169 vtss_phy_power_conf_t Struct Reference	334
7.169.1 Detailed Description	334
7.169.2 Field Documentation	334
7.169.2.1 mode	334
7.170 vtss_phy_power_status_t Struct Reference	335
7.170.1 Detailed Description	335
7.170.2 Field Documentation	335
7.170.2.1 level	335

7.171vtss_phy_reset_conf_t Struct Reference	335
7.171.1 Detailed Description	336
7.171.2 Field Documentation	336
7.171.2.1 mac_if	336
7.171.2.2 media_if	336
7.171.2.3 rgmii	336
7.171.2.4 tbi	336
7.171.2.5 force	337
7.171.2.6 pkt_mode	337
7.171.2.7 i_cpu_en	337
7.172vtss_phy_rgmii_conf_t Struct Reference	337
7.172.1 Detailed Description	337
7.172.2 Field Documentation	338
7.172.2.1 rx_clk_skew_ps	338
7.172.2.2 tx_clk_skew_ps	338
7.173vtss_phy_statistic_t Struct Reference	338
7.173.1 Detailed Description	338
7.173.2 Field Documentation	339
7.173.2.1 cu_good	339
7.173.2.2 cu_bad	339
7.173.2.3 serdes_tx_good	339
7.173.2.4 serdes_tx_bad	339
7.173.2.5 rx_err_cnt_base_tx	340
7.173.2.6 media_mac_serdes_good	340
7.173.2.7 media_mac_serdes_crc	340
7.174vtss_phy_status_1g_t Struct Reference	340
7.174.1 Detailed Description	341
7.174.2 Field Documentation	341
7.174.2.1 master_cfg_fault	341
7.174.2.2 master	341

7.175vtss_phy_tbi_conf_t Struct Reference	341
7.175.1 Detailed Description	341
7.175.2 Field Documentation	342
7.175.2.1 aneg_enable	342
7.176vtss_phy_timestamp_t Struct Reference	342
7.176.1 Detailed Description	342
7.176.2 Field Documentation	342
7.176.2.1 high	342
7.176.2.2 low	343
7.176.2.3 seconds	343
7.176.2.4 nanoseconds	343
7.177vtss_phy_ts_ach_conf_t Struct Reference	343
7.177.1 Detailed Description	344
7.177.2 Field Documentation	344
7.177.2.1 value [1/2]	344
7.177.2.2 mask [1/2]	344
7.177.2.3 version	344
7.177.2.4 value [2/2]	344
7.177.2.5 mask [2/2]	345
7.177.2.6 channel_type	345
7.177.2.7 proto_id	345
7.177.2.8 comm_opt	345
7.178vtss_phy_ts_alt_clock_mode_s Struct Reference	345
7.178.1 Detailed Description	346
7.178.2 Field Documentation	346
7.178.2.1 pps_ls_lpbk	346
7.178.2.2 ls_lpbk	346
7.178.2.3 ls_pps_lpbk	346
7.179vtss_phy_ts_eng_init_conf_t Struct Reference	346
7.179.1 Detailed Description	347

7.179.2 Field Documentation	347
7.179.2.1 eng_used	347
7.179.2.2 encaps_type	347
7.179.2.3 flow_match_mode	347
7.179.2.4 flow_st_index	348
7.179.2.5 flow_end_index	348
7.180vtss_phy_ts_engine_action_t Struct Reference	348
7.180.1 Detailed Description	348
7.180.2 Field Documentation	349
7.180.2.1 action_ptp	349
7.180.2.2 action_gen	349
7.180.2.3 ptp_conf	349
7.180.2.4 oam_conf	349
7.180.2.5 gen_conf	349
7.180.2.6 action	350
7.181vtss_phy_ts_engine_flow_conf_t Struct Reference	350
7.181.1 Detailed Description	350
7.181.2 Field Documentation	350
7.181.2.1 eng_mode	350
7.181.2.2 channel_map	351
7.181.2.3 ptp	351
7.181.2.4 oam	351
7.181.2.5 gen	351
7.181.2.6 flow_conf	351
7.182vtss_phy_ts_eth_conf_t Struct Reference	352
7.182.1 Detailed Description	352
7.182.2 Field Documentation	353
7.182.2.1 pbb_en	353
7.182.2.2 etype	353
7.182.2.3 tpid	353

7.182.2.4 comm_opt	353
7.182.2.5 flow_en	354
7.182.2.6 addr_match_mode	354
7.182.2.7 addr_match_select	354
7.182.2.8 mac_addr	354
7.182.2.9 vlan_check	354
7.182.2.10num_tag	355
7.182.2.11outer_tag_type	355
7.182.2.12inner_tag_type	355
7.182.2.13tag_range_mode	355
7.182.2.14upper	355
7.182.2.15lower	356
7.182.2.16range [1/2]	356
7.182.2.17val [1/2]	356
7.182.2.18mask [1/2]	356
7.182.2.19value [1/2]	356
7.182.2.20outer_tag	356
7.182.2.21range [2/2]	357
7.182.2.22value [2/2]	357
7.182.2.23val [2/2]	357
7.182.2.24mask [2/2]	357
7.182.2.25_tag	357
7.182.2.26inner_tag	357
7.182.2.27flow_opt	358
7.183vtss_phy_ts_fifo_conf_t Struct Reference	358
7.183.1 Detailed Description	358
7.183.2 Field Documentation	358
7.183.2.1 detect_only	358
7.183.2.2 eng_recov	359
7.183.2.3 eng_minE	359

7.183.2.4 skip_rev_check	359
7.184vtss_phy_ts_fifo_sig_t Struct Reference	359
7.184.1 Detailed Description	360
7.184.2 Field Documentation	360
7.184.2.1 sig_mask	360
7.184.2.2 msg_type	360
7.184.2.3 domain_num	360
7.184.2.4 src_port_identity	360
7.184.2.5 sequence_id	361
7.184.2.6 dest_ip	361
7.184.2.7 src_ip	361
7.184.2.8 dest_mac	361
7.185vtss_phy_ts_gen_conf_t Struct Reference	361
7.185.1 Detailed Description	362
7.185.2 Field Documentation	362
7.185.2.1 flow_offset	362
7.185.2.2 next_prot_offset	362
7.185.2.3 comm_opt	362
7.185.2.4 flow_en	363
7.185.2.5 data	363
7.185.2.6 mask	363
7.185.2.7 flow_opt	363
7.186vtss_phy_ts_generic_action_t Struct Reference	363
7.186.1 Detailed Description	364
7.186.2 Field Documentation	364
7.186.2.1 enable	364
7.186.2.2 channel_map	364
7.186.2.3 flow_id	364
7.186.2.4 data	365
7.186.2.5 mask	365

7.186.2.6 ts_type	365
7.186.2.7 ts_offset	365
7.187vtss_phy_ts_generic_flow_conf_t Struct Reference	365
7.187.1 Detailed Description	366
7.187.2 Field Documentation	366
7.187.2.1 eth1_opt	366
7.187.2.2 gen_opt	366
7.188vtss_phy_ts_ietf_mpls_ach_oam_conf_t Struct Reference	366
7.188.1 Detailed Description	367
7.188.2 Field Documentation	367
7.188.2.1 delaym_type	367
7.188.2.2 ts_format	367
7.188.2.3 ds	367
7.189vtss_phy_ts_init_conf_t Struct Reference	367
7.189.1 Detailed Description	368
7.189.2 Field Documentation	368
7.189.2.1 clk_freq	368
7.189.2.2 clk_src	368
7.189.2.3 rx_ts_pos	369
7.189.2.4 rx_ts_len	369
7.189.2.5 tx_fifo_mode	369
7.189.2.6 tx_ts_len	369
7.189.2.7 tx_fifo_spi_conf	369
7.189.2.8 tx_fifo_hi_clk_cycs	370
7.189.2.9 tx_fifo_lo_clk_cycs	370
7.189.2.10xaui_sel_8487	370
7.189.2.11tc_op_mode	370
7.189.2.12auto_clear_ls	370
7.189.2.13macsec_ena	371
7.189.2.14chk_ing_modified	371

7.189.2.15one_step_txfifo	371
7.190vtss_phy_ts_ip_conf_t Struct Reference	371
7.190.1 Detailed Description	372
7.190.2 Field Documentation	372
7.190.2.1 ip_mode	372
7.190.2.2 sport_val	373
7.190.2.3 sport_mask	373
7.190.2.4 dport_val	373
7.190.2.5 dport_mask	373
7.190.2.6 comm_opt	373
7.190.2.7 flow_en	374
7.190.2.8 match_mode	374
7.190.2.9 addr	374
7.190.2.10mask	374
7.190.2.11ipv4	374
7.190.2.12pv6	375
7.190.2.13p_addr	375
7.190.2.14flow_opt	375
7.191vtss_phy_ts_mpls_conf_t Struct Reference	375
7.191.1 Detailed Description	376
7.191.2 Field Documentation	376
7.191.2.1 cw_en	376
7.191.2.2 comm_opt	376
7.191.2.3 flow_en	376
7.191.2.4 stack_depth	376
7.191.2.5 stack_ref_point	377
7.191.2.6 top	377
7.191.2.7 frst_lvl_after_top	377
7.191.2.8 snd_lvl_after_top	377
7.191.2.9 thrd_lvl_after_top	377

7.191.2.10	top_down	378
7.191.2.11	end	378
7.191.2.12	rst_lvl_before_end	378
7.191.2.13	snd_lvl_before_end	378
7.191.2.14	hrd_lvl_before_end	378
7.191.2.15	bottom_up	378
7.191.2.16	stack_level	379
7.191.2.17	low_opt	379
7.192	vtss_phy_ts_mpls_lvl_rng_t Struct Reference	379
7.192.1	Detailed Description	379
7.192.2	Field Documentation	379
7.192.2.1	lower	379
7.192.2.2	upper	380
7.193	vtss_phy_ts_nphase_status_t Struct Reference	380
7.193.1	Detailed Description	380
7.193.2	Field Documentation	380
7.193.2.1	enable	380
7.193.2.2	CALIB_ERR	381
7.193.2.3	CALIB_DONE	381
7.194	vtss_phy_ts_oam_engine_action_t Struct Reference	381
7.194.1	Detailed Description	381
7.194.2	Field Documentation	382
7.194.2.1	enable	382
7.194.2.2	y1731_en	382
7.194.2.3	channel_map	382
7.194.2.4	version	382
7.194.2.5	y1731_oam_conf	382
7.194.2.6	ietf_oam_conf	383
7.194.2.7	oam_conf	383
7.195	vtss_phy_ts_oam_engine_flow_conf_t Struct Reference	383

7.195.1 Detailed Description	383
7.195.2 Field Documentation	383
7.195.2.1 eth1_opt	384
7.195.2.2 eth2_opt	384
7.195.2.3 mpls_opt	384
7.195.2.4 ach_opt	384
7.196vtss_phy_ts_pps_config_s Struct Reference	384
7.196.1 Detailed Description	385
7.196.2 Field Documentation	385
7.196.2.1 pps_width_adj	385
7.196.2.2 pps_offset	385
7.196.2.3 pps_output_enable	385
7.197vtss_phy_ts_ptp_conf_t Struct Reference	386
7.197.1 Detailed Description	386
7.197.2 Field Documentation	386
7.197.2.1 range_en	386
7.197.2.2 val	386
7.197.2.3 mask	387
7.197.2.4 value	387
7.197.2.5 upper	387
7.197.2.6 lower	387
7.197.2.7 range	387
7.197.2.8 domain	387
7.198vtss_phy_ts_ptp_engine_action_t Struct Reference	388
7.198.1 Detailed Description	388
7.198.2 Field Documentation	388
7.198.2.1 enable	388
7.198.2.2 channel_map	388
7.198.2.3 ptp_conf	389
7.198.2.4 clk_mode	389

7.198.2.5 <code>delaym_type</code>	389
7.198.2.6 <code>cf_update</code>	389
7.199 <code>vtss_phy_ts_ptp_engine_flow_conf_t</code> Struct Reference	389
7.199.1 Detailed Description	390
7.199.2 Field Documentation	390
7.199.2.1 <code>eth1_opt</code>	390
7.199.2.2 <code>eth2_opt</code>	390
7.199.2.3 <code>ip1_opt</code>	390
7.199.2.4 <code>ip2_opt</code>	391
7.199.2.5 <code>mpls_opt</code>	391
7.199.2.6 <code>ach_opt</code>	391
7.200 <code>vtss_phy_ts_sertod_conf_t</code> Struct Reference	391
7.200.1 Detailed Description	391
7.200.2 Field Documentation	392
7.200.2.1 <code>ip_enable</code>	392
7.200.2.2 <code>op_enable</code>	392
7.200.2.3 <code>ls_inv</code>	392
7.201 <code>vtss_phy_ts_stats_t</code> Struct Reference	392
7.201.1 Detailed Description	393
7.201.2 Field Documentation	393
7.201.2.1 <code>ingr_pream_shrink_err</code>	393
7.201.2.2 <code>egr_pream_shrink_err</code>	393
7.201.2.3 <code>ingr_fcs_err</code>	393
7.201.2.4 <code>egr_fcs_err</code>	394
7.201.2.5 <code>ingr_frm_mod_cnt</code>	394
7.201.2.6 <code>egr_frm_mod_cnt</code>	394
7.201.2.7 <code>ts_fifo_tx_cnt</code>	394
7.201.2.8 <code>ts_fifo_drop_cnt</code>	394
7.202 <code>vtss_phy_ts_y1731_oam_conf_t</code> Struct Reference	395
7.202.1 Detailed Description	395

7.202.2 Field Documentation	395
7.202.2.1 range_en	395
7.202.2.2 delaym_type	395
7.202.2.3 val	396
7.202.2.4 mask	396
7.202.2.5 value	396
7.202.2.6 upper	396
7.202.2.7 lower	396
7.202.2.8 range	396
7.202.2.9 meg_level	397
7.203vtss_phy_type_t Struct Reference	397
7.203.1 Detailed Description	397
7.203.2 Field Documentation	397
7.203.2.1 part_number	397
7.203.2.2 revision	398
7.203.2.3 port_cnt	398
7.203.2.4 channel_id	398
7.203.2.5 base_port_no	398
7.203.2.6 phy_api_base_no	398
7.204vtss_phy_veriphy_result_t Struct Reference	399
7.204.1 Detailed Description	399
7.204.2 Field Documentation	399
7.204.2.1 link	399
7.204.2.2 status	399
7.204.2.3 length	400
7.205vtss_phy_wol_conf_t Struct Reference	400
7.205.1 Detailed Description	400
7.205.2 Field Documentation	400
7.205.2.1 secure_on_enable	400
7.205.2.2 wol_mac	401

7.205.2.3 wol_pass	401
7.205.2.4 wol_passwd_len	401
7.205.2.5 magic_pkt_cnt	401
7.206vtss_pi_conf_t Struct Reference	401
7.206.1 Detailed Description	402
7.206.2 Field Documentation	402
7.206.2.1 cs_wait_ns	402
7.207vtss_policer_ext_t Struct Reference	402
7.207.1 Detailed Description	403
7.207.2 Field Documentation	403
7.207.2.1 frame_rate	403
7.207.2.2 dp_bypass_level	403
7.207.2.3 unicast	403
7.207.2.4 multicast	403
7.207.2.5 broadcast	404
7.207.2.6 uc_no_flood	404
7.207.2.7 mc_no_flood	404
7.207.2.8 flooded	404
7.207.2.9 learning	404
7.207.2.10o_cpu	405
7.207.2.11cpu_queue	405
7.207.2.12imit_noncpu_traffic	405
7.207.2.13imit_cpu_traffic	405
7.207.2.14flow_control	405
7.208vtss_policer_t Struct Reference	406
7.208.1 Detailed Description	406
7.208.2 Field Documentation	406
7.208.2.1 level	406
7.208.2.2 rate	406
7.209vtss_port_bridge_counters_t Struct Reference	406

7.209.1 Detailed Description	407
7.209.2 Field Documentation	407
7.209.2.1 dot1dTpPortInDiscards	407
7.210vtss_port_clause_37_adv_t Struct Reference	407
7.210.1 Detailed Description	407
7.210.2 Field Documentation	408
7.210.2.1 fdx	408
7.210.2.2 hdx	408
7.210.2.3 symmetric_pause	408
7.210.2.4 asymmetric_pause	408
7.210.2.5 remote_fault	408
7.210.2.6 acknowledge	409
7.210.2.7 next_page	409
7.211vtss_port_clause_37_control_t Struct Reference	409
7.211.1 Detailed Description	409
7.211.2 Field Documentation	409
7.211.2.1 enable	410
7.211.2.2 advertisement	410
7.212vtss_port_conf_t Struct Reference	410
7.212.1 Detailed Description	411
7.212.2 Field Documentation	411
7.212.2.1 if_type	411
7.212.2.2 sd_enable	411
7.212.2.3 sd_active_high	411
7.212.2.4 sd_internal	411
7.212.2.5 frame_gaps	412
7.212.2.6 power_down	412
7.212.2.7 speed	412
7.212.2.8 fdx	412
7.212.2.9 flow_control	412

7.212.2.10max_frame_length	413
7.212.2.11frame_length_chk	413
7.212.2.12max_tags	413
7.212.2.13exc_col_cont	413
7.212.2.14xaui_rx_lane_flip	413
7.212.2.15xaui_tx_lane_flip	414
7.212.2.16oop	414
7.212.2.17serdes	414
7.213vtss_port_counters_t Struct Reference	414
7.213.1 Detailed Description	415
7.213.2 Field Documentation	415
7.213.2.1 rmon	415
7.213.2.2 if_group	415
7.213.2.3 ethernet_like	415
7.213.2.4 bridge	415
7.213.2.5 prop	416
7.214vtss_port_ethernet_like_counters_t Struct Reference	416
7.214.1 Detailed Description	416
7.214.2 Field Documentation	416
7.214.2.1 dot3StatsAlignmentErrors	417
7.214.2.2 dot3StatsFCSErrors	417
7.214.2.3 dot3StatsFrameTooLongs	417
7.214.2.4 dot3StatsSymbolErrors	417
7.214.2.5 dot3ControlInUnknownOpcodes	417
7.214.2.6 dot3InPauseFrames	418
7.214.2.7 dot3StatsSingleCollisionFrames	418
7.214.2.8 dot3StatsMultipleCollisionFrames	418
7.214.2.9 dot3StatsDeferredTransmissions	418
7.214.2.10dot3StatsLateCollisions	418
7.214.2.11dot3StatsExcessiveCollisions	419

7.214.2.12dot3StatsCarrierSenseErrors	419
7.214.2.13dot3OutPauseFrames	419
7.215vtss_port_flow_control_conf_t Struct Reference	419
7.215.1 Detailed Description	420
7.215.2 Field Documentation	420
7.215.2.1 obey	420
7.215.2.2 generate	420
7.215.2.3 smac	420
7.215.2.4 pfc	420
7.216vtss_port_frame_gaps_t Struct Reference	421
7.216.1 Detailed Description	421
7.216.2 Field Documentation	421
7.216.2.1 hdx_gap_1	421
7.216.2.2 hdx_gap_2	421
7.216.2.3 fdx_gap	422
7.217vtss_port_if_group_counters_t Struct Reference	422
7.217.1 Detailed Description	422
7.217.2 Field Documentation	422
7.217.2.1 ifInOctets	423
7.217.2.2 ifInUcastPkts	423
7.217.2.3 ifInMulticastPkts	423
7.217.2.4 ifInBroadcastPkts	423
7.217.2.5 ifInNUcastPkts	423
7.217.2.6 ifInDiscards	424
7.217.2.7 ifInErrors	424
7.217.2.8 ifOutOctets	424
7.217.2.9 ifOutUcastPkts	424
7.217.2.10fOutMulticastPkts	424
7.217.2.11ifOutBroadcastPkts	425
7.217.2.12fOutNUcastPkts	425

7.217.2.13 fOutDiscards	425
7.217.2.14 fOutErrors	425
7.218 vtss_port_ifh_t Struct Reference	425
7.218.1 Detailed Description	426
7.218.2 Field Documentation	426
7.218.2.1 ena_ifh_header	426
7.219 vtss_port_map_t Struct Reference	426
7.219.1 Detailed Description	426
7.219.2 Field Documentation	427
7.219.2.1 chip_port	427
7.219.2.2 chip_no	427
7.219.2.3 miim_controller	427
7.219.2.4 miim_addr	427
7.219.2.5 miim_chip_no	428
7.220 vtss_port_proprietary_counters_t Struct Reference	428
7.220.1 Detailed Description	428
7.220.2 Field Documentation	428
7.220.2.1 rx_prio	428
7.220.2.2 tx_prio	429
7.221 vtss_port_rmon_counters_t Struct Reference	429
7.221.1 Detailed Description	430
7.221.2 Field Documentation	430
7.221.2.1 rx_etherStatsDropEvents	430
7.221.2.2 rx_etherStatsOctets	430
7.221.2.3 rx_etherStatsPkts	430
7.221.2.4 rx_etherStatsBroadcastPkts	430
7.221.2.5 rx_etherStatsMulticastPkts	431
7.221.2.6 rx_etherStatsCRCAlignErrors	431
7.221.2.7 rx_etherStatsUndersizePkts	431
7.221.2.8 rx_etherStatsOversizePkts	431

7.221.2.9 rx_etherStatsFragments	431
7.221.2.10rx_etherStatsJabbers	432
7.221.2.11rx_etherStatsPkts64Octets	432
7.221.2.12rx_etherStatsPkts65to127Octets	432
7.221.2.13rx_etherStatsPkts128to255Octets	432
7.221.2.14rx_etherStatsPkts256to511Octets	432
7.221.2.15rx_etherStatsPkts512to1023Octets	433
7.221.2.16rx_etherStatsPkts1024to1518Octets	433
7.221.2.17rx_etherStatsPkts1519toMaxOctets	433
7.221.2.18x_etherStatsDropEvents	433
7.221.2.19x_etherStatsOctets	433
7.221.2.20x_etherStatsPkts	434
7.221.2.21tx_etherStatsBroadcastPkts	434
7.221.2.22tx_etherStatsMulticastPkts	434
7.221.2.23tx_etherStatsCollisions	434
7.221.2.24tx_etherStatsPkts64Octets	434
7.221.2.25tx_etherStatsPkts65to127Octets	435
7.221.2.26tx_etherStatsPkts128to255Octets	435
7.221.2.27tx_etherStatsPkts256to511Octets	435
7.221.2.28tx_etherStatsPkts512to1023Octets	435
7.221.2.29tx_etherStatsPkts1024to1518Octets	435
7.221.2.30tx_etherStatsPkts1519toMaxOctets	436
7.222vtss_port_serdes_conf_t Struct Reference	436
7.222.1 Detailed Description	436
7.222.2 Field Documentation	436
7.222.2.1 sfp_dac	436
7.223vtss_port_sgmii_aneg_t Struct Reference	437
7.223.1 Detailed Description	437
7.223.2 Field Documentation	437
7.223.2.1 link	437

7.223.2.2 fdx	437
7.223.2.3 hdx	438
7.223.2.4 speed_10M	438
7.223.2.5 speed_100M	438
7.223.2.6 speed_1G	438
7.223.2.7 aneg_complete	438
7.224vtss_port_status_t Struct Reference	439
7.224.1 Detailed Description	439
7.224.2 Field Documentation	439
7.224.2.1 link_down	439
7.224.2.2 link	439
7.224.2.3 speed	440
7.224.2.4 fdx	440
7.224.2.5 remote_fault	440
7.224.2.6 aneg_complete	440
7.224.2.7 unidirectional_ability	440
7.224.2.8 aneg	441
7.224.2.9 mdi_cross	441
7.224.2.10fiber	441
7.224.2.11copper	441
7.225vtss_qce_action_t Struct Reference	441
7.225.1 Detailed Description	442
7.225.2 Field Documentation	442
7.225.2.1 prio_enable	442
7.225.2.2 prio	442
7.225.2.3 dp_enable	442
7.225.2.4 dp	443
7.225.2.5 dscp_enable	443
7.225.2.6 dscp	443
7.226vtss_qce_frame_etype_t Struct Reference	443

7.226.1 Detailed Description	443
7.226.2 Field Documentation	444
7.226.2.1 <code>etype</code>	444
7.226.2.2 <code>data</code>	444
7.227 <code>vtss_qce_frame_ipv4_t</code> Struct Reference	444
7.227.1 Detailed Description	444
7.227.2 Field Documentation	445
7.227.2.1 <code>fragment</code>	445
7.227.2.2 <code>dscp</code>	445
7.227.2.3 <code>proto</code>	445
7.227.2.4 <code>sip</code>	445
7.227.2.5 <code>sport</code>	445
7.227.2.6 <code>dport</code>	446
7.228 <code>vtss_qce_frame_ipv6_t</code> Struct Reference	446
7.228.1 Detailed Description	446
7.228.2 Field Documentation	446
7.228.2.1 <code>dscp</code>	446
7.228.2.2 <code>proto</code>	447
7.228.2.3 <code>sip</code>	447
7.228.2.4 <code>sport</code>	447
7.228.2.5 <code>dport</code>	447
7.229 <code>vtss_qce_frame_llc_t</code> Struct Reference	447
7.229.1 Detailed Description	448
7.229.2 Field Documentation	448
7.229.2.1 <code>data</code>	448
7.230 <code>vtss_qce_frame_snap_t</code> Struct Reference	448
7.230.1 Detailed Description	448
7.230.2 Field Documentation	448
7.230.2.1 <code>data</code>	449
7.231 <code>vtss_qce_key_t</code> Struct Reference	449

7.231.1 Detailed Description	449
7.231.2 Field Documentation	449
7.231.2.1 port_list	449
7.231.2.2 mac	450
7.231.2.3 tag	450
7.231.2.4 type	450
7.231.2.5 etype	450
7.231.2.6 llc	450
7.231.2.7 snap	451
7.231.2.8 ipv4	451
7.231.2.9 ipv6	451
7.231.2.10 frame	451
7.232vtss_qce_mac_t Struct Reference	451
7.232.1 Detailed Description	452
7.232.2 Field Documentation	452
7.232.2.1 dmac_mc	452
7.232.2.2 dmac_bc	452
7.232.2.3 smac	452
7.233vtss_qce_t Struct Reference	453
7.233.1 Detailed Description	453
7.233.2 Field Documentation	453
7.233.2.1 id	453
7.233.2.2 key	453
7.233.2.3 action	454
7.234vtss_qce_tag_t Struct Reference	454
7.234.1 Detailed Description	454
7.234.2 Field Documentation	454
7.234.2.1 vid	454
7.234.2.2 pcp	455
7.234.2.3 dei	455

7.234.2.4 tagged	455
7.235vtss_qos_conf_t Struct Reference	455
7.235.1 Detailed Description	456
7.235.2 Field Documentation	456
7.235.2.1 prios	456
7.235.2.2 dscp_trust	456
7.235.2.3 dscp_qos_class_map	456
7.235.2.4 dscp_dp_level_map	456
7.235.2.5 dscp_qos_map	457
7.235.2.6 dscp_remark	457
7.235.2.7 dscp_translate_map	457
7.235.2.8 dscp_remap	457
7.236vtss_qos_port_conf_t Struct Reference	457
7.236.1 Detailed Description	458
7.236.2 Field Documentation	458
7.236.2.1 red	458
7.236.2.2 policer_port	459
7.236.2.3 policer_ext_port	459
7.236.2.4 policer_queue	459
7.236.2.5 shaper_port	459
7.236.2.6 shaper_queue	459
7.236.2.7 excess_enable	460
7.236.2.8 default_prio	460
7.236.2.9 usr_prio	460
7.236.2.10 default_dpl	460
7.236.2.11 default_dei	460
7.236.2.12 ag_class_enable	461
7.236.2.13 qos_class_map	461
7.236.2.14 dp_level_map	461
7.236.2.15 dscp_class_enable	461

7.236.2.16dscp_mode	461
7.236.2.17dscp_emode	462
7.236.2.18dscp_translate	462
7.236.2.19ag_remark_mode	462
7.236.2.20tag_default_pcp	462
7.236.2.21tag_default_dei	462
7.236.2.22tag_pcp_map	463
7.236.2.23tag_dei_map	463
7.236.2.24dwrr_enable	463
7.236.2.25queue_pct	463
7.237vtss_qs_conf_t Struct Reference	463
7.237.1 Detailed Description	464
7.237.2 Field Documentation	464
7.237.2.1 mode	464
7.237.2.2 oversubscription	464
7.237.2.3 port_min	464
7.237.2.4 port_max	465
7.237.2.5 queue_min	465
7.237.2.6 queue_max	465
7.237.2.7 port	465
7.238vtss_rcpll_status_t Struct Reference	465
7.238.1 Detailed Description	466
7.238.2 Field Documentation	466
7.238.2.1 out_of_range	466
7.238.2.2 cal_error	466
7.238.2.3 cal_not_done	466
7.239vtss_red_t Struct Reference	467
7.239.1 Detailed Description	467
7.239.2 Field Documentation	467
7.239.2.1 enable	467

7.239.2.2 max_th	467
7.239.2.3 min_th	468
7.239.2.4 max_prob_1	468
7.239.2.5 max_prob_2	468
7.239.2.6 max_prob_3	468
7.240vtss_restart_status_t Struct Reference	468
7.240.1 Detailed Description	469
7.240.2 Field Documentation	469
7.240.2.1 restart	469
7.240.2.2 prev_version	469
7.240.2.3 cur_version	469
7.241vtss_routing_entry_t Struct Reference	470
7.241.1 Detailed Description	470
7.241.2 Field Documentation	470
7.241.2.1 type	470
7.241.2.2 ipv4_uc	470
7.241.2.3 ipv6_uc	471
7.241.2.4 route	471
7.241.2.5 vlan	471
7.242vtss_secure_on_passwd_t Struct Reference	471
7.242.1 Detailed Description	471
7.242.2 Field Documentation	472
7.242.2.1 passwd	472
7.243vtss_serd़es_macro_conf_t Struct Reference	472
7.243.1 Detailed Description	472
7.243.2 Field Documentation	472
7.243.2.1 serdes1g_vdd	472
7.243.2.2 serdes6g_vdd	473
7.243.2.3 ib_cterm_ena	473
7.243.2.4 qsgmii	473

7.244vtss_sflow_port_conf_t Struct Reference	473
7.244.1 Detailed Description	474
7.244.2 Field Documentation	474
7.244.2.1 type	474
7.244.2.2 sampling_rate	474
7.245vtss_sgpi_conf_t Struct Reference	474
7.245.1 Detailed Description	475
7.245.2 Field Documentation	475
7.245.2.1 bmode	475
7.245.2.2 bit_count	475
7.245.2.3 port_conf	475
7.246vtss_sgpi_port_conf_t Struct Reference	475
7.246.1 Detailed Description	476
7.246.2 Field Documentation	476
7.246.2.1 enabled	476
7.246.2.2 mode	476
7.246.2.3 int_pol_high	476
7.247vtss_sgpi_port_data_t Struct Reference	477
7.247.1 Detailed Description	477
7.247.2 Field Documentation	477
7.247.2.1 value	477
7.248vtss_shaper_t Struct Reference	477
7.248.1 Detailed Description	478
7.248.2 Field Documentation	478
7.248.2.1 level	478
7.248.2.2 rate	478
7.249vtss_sublayer_status_t Struct Reference	478
7.249.1 Detailed Description	479
7.249.2 Field Documentation	479
7.249.2.1 rx_link	479

7.249.2.2 link_down	479
7.249.2.3 rx_fault	479
7.249.2.4 tx_fault	479
7.250vtss_sync_clock_in_t Struct Reference	480
7.250.1 Detailed Description	480
7.250.2 Field Documentation	480
7.250.2.1 port_no	480
7.250.2.2 squelsh	480
7.250.2.3 enable	481
7.251vtss_sync_clock_out_t Struct Reference	481
7.251.1 Detailed Description	481
7.251.2 Field Documentation	481
7.251.2.1 divider	481
7.251.2.2 enable	482
7.252vtss_tci_t Struct Reference	482
7.252.1 Detailed Description	482
7.252.2 Field Documentation	482
7.252.2.1 vid	482
7.252.2.2 cfi	483
7.252.2.3 tagprior	483
7.253vtss_timeofday_t Struct Reference	483
7.253.1 Detailed Description	483
7.253.2 Field Documentation	483
7.253.2.1 sec [1/2]	484
7.253.2.2 sec [2/2]	484
7.254vtss_timestamp_t Struct Reference	484
7.254.1 Detailed Description	484
7.254.2 Field Documentation	484
7.254.2.1 sec_msb	485
7.254.2.2 seconds	485

7.254.2.3 nanoseconds	485
7.255vtss_trace_conf_t Struct Reference	485
7.255.1 Detailed Description	485
7.255.2 Field Documentation	486
7.255.2.1 level	486
7.256vtss_ts_ext_clock_mode_t Struct Reference	486
7.256.1 Detailed Description	486
7.256.2 Field Documentation	486
7.256.2.1 one_pps_mode	486
7.256.2.2 enable	487
7.256.2.3 freq	487
7.257vtss_ts_id_t Struct Reference	487
7.257.1 Detailed Description	487
7.257.2 Field Documentation	487
7.257.2.1 ts_id	488
7.258vtss_ts_internal_mode_t Struct Reference	488
7.258.1 Detailed Description	488
7.258.2 Field Documentation	488
7.258.2.1 int_fmt	488
7.259vtss_ts_operation_mode_t Struct Reference	489
7.259.1 Detailed Description	489
7.259.2 Field Documentation	489
7.259.2.1 mode	489
7.260vtss_ts_timestamp_alloc_t Struct Reference	489
7.260.1 Detailed Description	490
7.260.2 Field Documentation	490
7.260.2.1 port_mask	490
7.260.2.2 context	490
7.260.2.3 cb	490
7.261vtss_ts_timestamp_t Struct Reference	490

7.261.1 Detailed Description	491
7.261.2 Field Documentation	491
7.261.2.1 ts	491
7.261.2.2 id	491
7.261.2.3 context	491
7.261.2.4 ts_valid	492
7.262vtss_vcap_ip_t Struct Reference	492
7.262.1 Detailed Description	492
7.262.2 Field Documentation	492
7.262.2.1 value	492
7.262.2.2 mask	493
7.263vtss_vcap_u128_t Struct Reference	493
7.263.1 Detailed Description	493
7.263.2 Field Documentation	493
7.263.2.1 value	493
7.263.2.2 mask	494
7.264vtss_vcap_u16_t Struct Reference	494
7.264.1 Detailed Description	494
7.264.2 Field Documentation	494
7.264.2.1 value	494
7.264.2.2 mask	495
7.265vtss_vcap_u24_t Struct Reference	495
7.265.1 Detailed Description	495
7.265.2 Field Documentation	495
7.265.2.1 value	495
7.265.2.2 mask	496
7.266vtss_vcap_u32_t Struct Reference	496
7.266.1 Detailed Description	496
7.266.2 Field Documentation	496
7.266.2.1 value	496

7.266.2.2 mask	497
7.267vtss_vcap_u40_t Struct Reference	497
7.267.1 Detailed Description	497
7.267.2 Field Documentation	497
7.267.2.1 value	497
7.267.2.2 mask	498
7.268vtss_vcap_u48_t Struct Reference	498
7.268.1 Detailed Description	498
7.268.2 Field Documentation	498
7.268.2.1 value	498
7.268.2.2 mask	499
7.269vtss_vcap_u8_t Struct Reference	499
7.269.1 Detailed Description	499
7.269.2 Field Documentation	499
7.269.2.1 value	499
7.269.2.2 mask	500
7.270vtss_vcap_udp_tcp_t Struct Reference	500
7.270.1 Detailed Description	500
7.270.2 Field Documentation	500
7.270.2.1 in_range	500
7.270.2.2 low	501
7.270.2.3 high	501
7.271vtss_vcap_vid_t Struct Reference	501
7.271.1 Detailed Description	501
7.271.2 Field Documentation	501
7.271.2.1 value	502
7.271.2.2 mask	502
7.272vtss_vcap_vr_t Struct Reference	502
7.272.1 Detailed Description	502
7.272.2 Field Documentation	503

7.272.2.1 type	503
7.272.2.2 value	503
7.272.2.3 mask	503
7.272.2.4 v	503
7.272.2.5 low	503
7.272.2.6 high	504
7.272.2.7 r	504
7.272.2.8 vr	504
7.273vtss_vce_action_t Struct Reference	504
7.273.1 Detailed Description	504
7.273.2 Field Documentation	504
7.273.2.1 vid	505
7.273.2.2 policy_no	505
7.274vtss_vce_frame_etype_t Struct Reference	505
7.274.1 Detailed Description	505
7.274.2 Field Documentation	505
7.274.2.1 etype	506
7.274.2.2 data	506
7.275vtss_vce_frame_ipv4_t Struct Reference	506
7.275.1 Detailed Description	506
7.275.2 Field Documentation	506
7.275.2.1 fragment	507
7.275.2.2 options	507
7.275.2.3 dscp	507
7.275.2.4 proto	507
7.275.2.5 sip	507
7.275.2.6 dport	508
7.276vtss_vce_frame_ipv6_t Struct Reference	508
7.276.1 Detailed Description	508
7.276.2 Field Documentation	508

7.276.2.1 dscp	508
7.276.2.2 proto	509
7.276.2.3 sip	509
7.276.2.4 dport	509
7.277vtss_vce_frame_llc_t Struct Reference	509
7.277.1 Detailed Description	509
7.277.2 Field Documentation	510
7.277.2.1 data	510
7.278vtss_vce_frame_snap_t Struct Reference	510
7.278.1 Detailed Description	510
7.278.2 Field Documentation	510
7.278.2.1 data	510
7.279vtss_vce_key_t Struct Reference	511
7.279.1 Detailed Description	511
7.279.2 Field Documentation	511
7.279.2.1 port_list	511
7.279.2.2 mac	511
7.279.2.3 tag	512
7.279.2.4 type	512
7.279.2.5 etype	512
7.279.2.6 llc	512
7.279.2.7 snap	512
7.279.2.8 ipv4	513
7.279.2.9 ipv6	513
7.279.2.10 frame	513
7.280vtss_vce_mac_t Struct Reference	513
7.280.1 Detailed Description	513
7.280.2 Field Documentation	514
7.280.2.1 dmac_mc	514
7.280.2.2 dmac_bc	514

7.280.2.3 smac	514
7.281vtss_vce_t Struct Reference	514
7.281.1 Detailed Description	515
7.281.2 Field Documentation	515
7.281.2.1 id	515
7.281.2.2 key	515
7.281.2.3 action	515
7.282vtss_vce_tag_t Struct Reference	515
7.282.1 Detailed Description	516
7.282.2 Field Documentation	516
7.282.2.1 vid	516
7.282.2.2 pcp	516
7.282.2.3 dei	516
7.282.2.4 tagged	517
7.282.2.5 s_tag	517
7.283vtss_vcl_port_conf_t Struct Reference	517
7.283.1 Detailed Description	517
7.283.2 Field Documentation	517
7.283.2.1 dmac_dip	518
7.284vtss_vid_mac_t Struct Reference	518
7.284.1 Detailed Description	518
7.284.2 Field Documentation	518
7.284.2.1 vid	518
7.284.2.2 mac	519
7.285vtss_vlan_conf_t Struct Reference	519
7.285.1 Detailed Description	519
7.285.2 Field Documentation	519
7.285.2.1 s_etype	519
7.286vtss_vlan_port_conf_t Struct Reference	520
7.286.1 Detailed Description	520

7.286.2 Field Documentation	520
7.286.2.1 port_type	520
7.286.2.2 pvid	520
7.286.2.3 untagged_vid	521
7.286.2.4 frame_type	521
7.286.2.5 ingress_filter	521
7.287vtss_vlan_tag_t Struct Reference	521
7.287.1 Detailed Description	521
7.287.2 Field Documentation	522
7.287.2.1 tpid	522
7.287.2.2 pcp	522
7.287.2.3 dei	522
7.287.2.4 vid	522
7.288vtss_vlan_trans_grp2vlan_conf_t Struct Reference	523
7.288.1 Detailed Description	523
7.288.2 Field Documentation	523
7.288.2.1 group_id	523
7.288.2.2 vid	523
7.288.2.3 trans_vid	524
7.289vtss_vlan_trans_port2grp_conf_t Struct Reference	524
7.289.1 Detailed Description	524
7.289.2 Field Documentation	524
7.289.2.1 group_id	524
7.289.2.2 ports	525
7.290vtss_vlan_vid_conf_t Struct Reference	525
7.290.1 Detailed Description	525
7.290.2 Field Documentation	525
7.290.2.1 learning	525
7.290.2.2 mirror	526
7.290.2.3 fid	526

7.291vtss_vstax_conf_t Struct Reference	526
7.291.1 Detailed Description	526
7.291.2 Field Documentation	526
7.291.2.1 upsid_0	527
7.291.2.2 upsid_1	527
7.291.2.3 port_0	527
7.291.2.4 port_1	527
7.291.2.5 cmef_disable	527
7.292vtss_vstax_glag_entry_t Struct Reference	528
7.292.1 Detailed Description	528
7.292.2 Field Documentation	528
7.292.2.1 upsid	528
7.292.2.2 upspn	528
7.293vtss_vstax_port_conf_t Struct Reference	528
7.293.1 Detailed Description	529
7.293.2 Field Documentation	529
7.293.2.1 ttl	529
7.293.2.2 mirror	529
7.294vtss_vstax_route_entry_t Struct Reference	529
7.294.1 Detailed Description	530
7.294.2 Field Documentation	530
7.294.2.1 stack_port_a	530
7.294.2.2 stack_port_b	530
7.295vtss_vstax_route_table_t Struct Reference	530
7.295.1 Detailed Description	531
7.295.2 Field Documentation	531
7.295.2.1 topology_type	531
7.295.2.2 table	531
7.296vtss_vstax_rx_header_t Struct Reference	531
7.296.1 Detailed Description	532

7.296.2 Field Documentation	532
7.296.2.1 valid	532
7.296.2.2 sp	532
7.296.2.3 upsid	532
7.296.2.4 port_no	532
7.296.2.5 glag_no	533
7.296.2.6 isdx	533
7.297vtss_vstax_tx_header_t Struct Reference	533
7.297.1 Detailed Description	533
7.297.2 Field Documentation	534
7.297.2.1 fwd_mode	534
7.297.2.2 ttl	534
7.297.2.3 prio	534
7.297.2.4 upsid	534
7.297.2.5 tci	534
7.297.2.6 port_no	535
7.297.2.7 chip_port	535
7.297.2.8 glag_no	535
7.297.2.9 queue_no	535
7.297.2.10keep_ttl	535
7.297.2.11dp	536
7.298vtss_wol_mac_addr_t Struct Reference	536
7.298.1 Detailed Description	536
7.298.2 Field Documentation	536
7.298.2.1 addr	536

8 File Documentation	537
8.1 vtss_api/include/vtss/api/I2_types.h File Reference	537
8.1.1 Detailed Description	537
8.1.2 Enumeration Type Documentation	537
8.1.2.1 vtss_sflow_type_t	537
8.2 vtss_api/include/vtss/api/options.h File Reference	538
8.2.1 Detailed Description	539
8.2.2 Macro Definition Documentation	540
8.2.2.1 VTSS_ARCH_JAGUAR_1	540
8.2.2.2 VTSS_ARCH_JAGUAR_1_CE_SWITCH	540
8.2.2.3 VTSS_FEATURE_EVC	540
8.2.2.4 VTSS_FEATURE_E_TREE	540
8.2.2.5 VTSS_FEATURE_TIMESTAMP	540
8.2.2.6 VTSS_FEATURE_PHY_TIMESTAMP	541
8.2.2.7 VTSS_FEATURE_VSTAX	541
8.2.2.8 VTSS_FEATURE_AGGR_GLAG	541
8.2.2.9 VTSS_FEATURE_VSTAX_V2	541
8.2.2.10 VTSS_PHY_TS_SPI_CLK_THRU_PPS0	541
8.2.2.11 VTSS_FEATURE_FAN [1/2]	542
8.2.2.12 VTSS_FEATURE_PVLAN	542
8.2.2.13 VTSS_OPT_TS_SPI_FPGA	542
8.2.2.14 VTSS_CHIP_10G_PHY	542
8.2.2.15 VTSS_FEATURE_MISC	542
8.2.2.16 VTSS_FEATURE_SERIAL_GPIO	543
8.2.2.17 VTSS_FEATURE_PORT_CONTROL	543
8.2.2.18 VTSS_FEATURE_PORT_IFH	543
8.2.2.19 VTSS_FEATURE_CLAUSE_37	543
8.2.2.20 VTSS_FEATURE_10G	543
8.2.2.21 VTSS_FEATURE_NPI	544
8.2.2.22 VTSS_FEATURE_EXC_COL_CONT	544

8.2.2.23	VTSS_FEATURE_PORT_CNT_ETHER_LIKE	544
8.2.2.24	VTSS_FEATURE_PORT_CNT_BRIDGE	544
8.2.2.25	VTSS_FEATURE_QOS	544
8.2.2.26	VTSS_FEATURE_QCL	545
8.2.2.27	VTSS_FEATURE_QCL_V2	545
8.2.2.28	VTSS_FEATURE_QOS_PORT_POLICER_EXT	545
8.2.2.29	VTSS_FEATURE_QOS_PORT_POLICER_EXT_FPS	545
8.2.2.30	VTSS_FEATURE_QOS_PORT_POLICER_EXT_FC	545
8.2.2.31	VTSS_FEATURE_QOS_PORT_POLICER_EXT_DPBL	546
8.2.2.32	VTSS_FEATURE_QOS_PORT_POLICER_EXT_TTM	546
8.2.2.33	VTSS_FEATURE_QOS_QUEUE_POLICER	546
8.2.2.34	VTSS_FEATURE_QOS_QUEUE_TX	546
8.2.2.35	VTSS_FEATURE_QOS_SCHEDULER_V2	546
8.2.2.36	VTSS_FEATURE_QOS_TAG_REMARK_V2	547
8.2.2.37	VTSS_FEATURE_QOS_CLASSIFICATION_V2	547
8.2.2.38	VTSS_FEATURE_QOS_EGRESS_QUEUE_SHAPERS	547
8.2.2.39	VTSS_FEATURE_QOS_EGRESS_QUEUE_SHAPERS_EB	547
8.2.2.40	VTSS_FEATURE_QOS_DSCP_REMARK	547
8.2.2.41	VTSS_FEATURE_QOS_DSCP_REMARK_V2	548
8.2.2.42	VTSS_FEATURE_QOS_WRED	548
8.2.2.43	VTSS_FEATURE_PACKET	548
8.2.2.44	VTSS_FEATURE_PACKET_TX	548
8.2.2.45	VTSS_FEATURE_PACKET_RX	548
8.2.2.46	VTSS_FEATURE_PACKET_GROUPING	549
8.2.2.47	VTSS_FEATURE_PACKET_PORT_REG	549
8.2.2.48	VTSS_FEATURE_LAYER2	549
8.2.2.49	VTSS_FEATURE_VLAN_PORT_V2	549
8.2.2.50	VTSS_FEATURE_VLAN_TX_TAG	549
8.2.2.51	VTSS_FEATURE_VLAN_SVL	550
8.2.2.52	VTSS_FEATURE_MAC_AGE_AUTO	550

8.2.2.53	VTSS_FEATURE_MAC_CPU_QUEUE	550
8.2.2.54	VTSS_FEATURE_LAYER3	550
8.2.2.55	VTSS_FEATURE_PORT_MUX	550
8.2.2.56	VTSS_FEATURE_VCAP [1/2]	551
8.2.2.57	VTSS_FEATURE_ACL	551
8.2.2.58	VTSS_FEATURE_ACL_V1	551
8.2.2.59	VTSS_FEATURE_VCL	551
8.2.2.60	VTSS_FEATURE_SYNCE	551
8.2.2.61	VTSS_FEATURE_FAN [2/2]	552
8.2.2.62	VTSS_FEATURE_EEE	552
8.2.2.63	VTSS_FEATURE_SERDES_MACRO_SETTINGS	552
8.2.2.64	VTSS_FEATURE_LED_POW_REDUC	552
8.2.2.65	VTSS_FEATURE_10GBASE_KR	552
8.2.2.66	VTSS_FEATURE_IRQ_CONTROL	553
8.2.2.67	VTSS_FEATURE_VLAN_TRANSLATION	553
8.2.2.68	VTSS_FEATURE_SFLOW	553
8.2.2.69	VTSS_PHY_10G_FIFO_SYNC	553
8.2.2.70	VIPER_B_FIFO_RESET	553
8.2.2.71	VTSS_FEATURE_QOS_POLICER_DL8	554
8.2.2.72	VTSS_CHIP CU PHY	554
8.2.2.73	VTSS_OPT_TRACE	554
8.2.2.74	VTSS_OPT_VAUI_EQ_CTRL	554
8.2.2.75	VTSS_PHY_OPT_VERIPHYS	554
8.2.2.76	VTSS_FEATURE_WARM_START [1/2]	555
8.2.2.77	VTSS_FEATURE_SYNCE_10G	555
8.2.2.78	VTSS_FEATURE_EDC_FW_LOAD	555
8.2.2.79	VTSS_FEATURE_WIS	555
8.2.2.80	VTSS_FEATURE_WARM_START [2/2]	555
8.2.2.81	VTSS_ARCH_MALIBU	556
8.2.2.82	VTSS_ARCH_MALIBU_B	556

8.2.2.83	VTSS_ARCH_VENICE_C	556
8.2.2.84	VTSS_FEATURE_VCAP [2/2]	556
8.3	vtss_api/include/vtss/api/phy.h File Reference	556
8.3.1	Detailed Description	557
8.3.2	Macro Definition Documentation	557
8.3.2.1	VTSS_PHY_POWER_ACTIPHY_BIT	557
8.3.2.2	VTSS_PHY_POWER_DYNAMIC_BIT	557
8.3.3	Enumeration Type Documentation	557
8.3.3.1	vtss_phy_power_mode_t	557
8.3.3.2	vtss_phy_veriphy_status_t	558
8.4	vtss_api/include/vtss/api/port.h File Reference	558
8.4.1	Detailed Description	560
8.4.2	Macro Definition Documentation	561
8.4.2.1	PORT_CAP_NONE	561
8.4.2.2	PORT_CAP_AUTONEG	561
8.4.2.3	PORT_CAP_10M_HDX	561
8.4.2.4	PORT_CAP_10M_FDX	561
8.4.2.5	PORT_CAP_100M_HDX	561
8.4.2.6	PORT_CAP_100M_FDX	562
8.4.2.7	PORT_CAP_1G_FDX	562
8.4.2.8	PORT_CAP_2_5G_FDX	562
8.4.2.9	PORT_CAP_5G_FDX	562
8.4.2.10	PORT_CAP_10G_FDX	562
8.4.2.11	PORT_CAP_FLOW_CTRL	563
8.4.2.12	PORT_CAP_COPPER	563
8.4.2.13	PORT_CAP_FIBER	563
8.4.2.14	PORT_CAP_DUAL_COPPER	563
8.4.2.15	PORT_CAP_DUAL_FIBER	563
8.4.2.16	PORT_CAP_SD_ENABLE	564
8.4.2.17	PORT_CAP_SD_HIGH	564

8.4.2.18	PORT_CAP_SD_INTERNAL	564
8.4.2.19	PORT_CAP_DUAL_FIBER_100FX	564
8.4.2.20	PORT_CAP_XAUI_LANE_FLIP	564
8.4.2.21	PORT_CAP_VTSS_10G_PHY	565
8.4.2.22	PORT_CAP_SFP_DETECT	565
8.4.2.23	PORT_CAP_STACKING	565
8.4.2.24	PORT_CAP_DUAL_SFP_DETECT	565
8.4.2.25	PORT_CAP_SFP_ONLY	565
8.4.2.26	PORT_CAP_DUAL_COPPER_100FX	566
8.4.2.27	PORT_CAP_HDX	566
8.4.2.28	PORT_CAP_TRI_SPEED_FDX	566
8.4.2.29	PORT_CAP_TRI_SPEED	566
8.4.2.30	PORT_CAP_1G_PHY	566
8.4.2.31	PORT_CAP_TRI_SPEED_COPPER	567
8.4.2.32	PORT_CAP_TRI_SPEED_FIBER	567
8.4.2.33	PORT_CAP_TRI_SPEED_DUAL_COPPER	567
8.4.2.34	PORT_CAP_TRI_SPEED_DUAL_FIBER	567
8.4.2.35	PORT_CAP_TRI_SPEED_DUAL_FIBER_100FX	567
8.4.2.36	PORT_CAP_ANY_FIBER	568
8.4.2.37	PORT_CAP_SPEED_DUAL_ANY_FIBER_FIXED_SPEED	568
8.4.2.38	PORT_CAP_SPEED_DUAL_ANY_FIBER	568
8.4.2.39	PORT_CAP_TRI_SPEED_DUAL_ANY_FIBER	568
8.4.2.40	PORT_CAP_TRI_SPEED_DUAL_ANY_FIBER_FIXED_SFP_SPEED	568
8.4.2.41	PORT_CAP_DUAL_FIBER_1000X	569
8.4.2.42	PORT_CAP_SFP_1G	569
8.4.2.43	PORT_CAP_SFP_2_5G	569
8.4.2.44	PORT_CAP_SFP_SD_HIGH	569
8.4.2.45	PORT_CAP_2_5G_TRI_SPEED_FDX	569
8.4.2.46	PORT_CAP_2_5G_TRI_SPEED	570
8.4.2.47	PORT_CAP_2_5G_TRI_SPEED_COPPER	570

8.4.3	Typedef Documentation	570
8.4.3.1	port_cap_t	570
8.4.4	Enumeration Type Documentation	570
8.4.4.1	vtss_port_speed_t	570
8.4.4.2	vtss_fiber_port_speed_t	571
8.5	vtss_api/include/vtss/api/types.h File Reference	571
8.5.1	Detailed Description	578
8.5.2	Macro Definition Documentation	578
8.5.2.1	PRIu64	579
8.5.2.2	PRIi64	579
8.5.2.3	PRIx64	579
8.5.2.4	VTSS_BIT64	579
8.5.2.5	VTSS_BITMASK64	579
8.5.2.6	VTSS_EXTRACT_BITFIELD64	580
8.5.2.7	VTSS_ENCODE_BITFIELD64	580
8.5.2.8	VTSS_ENCODE_BITMASK64	580
8.5.2.9	TRUE	580
8.5.2.10	FALSE	581
8.5.2.11	VTSS_PACKET_RATE_DISABLED	581
8.5.2.12	VTSS_PORT_COUNT [1/2]	581
8.5.2.13	VTSS_PORT_COUNT [2/2]	581
8.5.2.14	VTSS_PORTS	581
8.5.2.15	VTSS_PORT_NO_NONE	582
8.5.2.16	VTSS_PORT_NO_CPU	582
8.5.2.17	VTSS_PORT_NO_START	582
8.5.2.18	VTSS_PORT_NO_END	582
8.5.2.19	VTSS_PORT_ARRAY_SIZE	582
8.5.2.20	VTSS_PORT_IS_PORT	583
8.5.2.21	VTSS_PRIOS	583
8.5.2.22	VTSS_PRIO_NO_NONE	583

8.5.2.23	VTSS_PRIO_START	583
8.5.2.24	VTSS_PRIO_END	583
8.5.2.25	VTSS_PRIO_ARRAY_SIZE	584
8.5.2.26	VTSS_QUEUES	584
8.5.2.27	VTSS_QUEUE_START	584
8.5.2.28	VTSS_QUEUE_END	584
8.5.2.29	VTSS_QUEUE_ARRAY_SIZE	584
8.5.2.30	VTSS_PCPS	585
8.5.2.31	VTSS_PCP_START	585
8.5.2.32	VTSS_PCP_END	585
8.5.2.33	VTSS_PCP_ARRAY_SIZE	585
8.5.2.34	VTSS_DEIS	585
8.5.2.35	VTSS_DEI_START	586
8.5.2.36	VTSS_DEI_END	586
8.5.2.37	VTSS_DEI_ARRAY_SIZE	586
8.5.2.38	VTSS_DPLS [1/2]	586
8.5.2.39	VTSS_DPLS [2/2]	586
8.5.2.40	VTSS_DPL_START	587
8.5.2.41	VTSS_DPL_END	587
8.5.2.42	VTSS_DPL_ARRAY_SIZE	587
8.5.2.43	VTSS_BITRATE_DISABLED	587
8.5.2.44	VTSS_VID_NULL	587
8.5.2.45	VTSS_VID_DEFAULT	588
8.5.2.46	VTSS_VID_RESERVED	588
8.5.2.47	VTSS_VIDS	588
8.5.2.48	VTSS_VID_ALL	588
8.5.2.49	VTSSETYPE_VTSS	588
8.5.2.50	VTSS_MAC_ADDR_SZ_BYTES	589
8.5.2.51	MAC_ADDR_BROADCAST	589
8.5.2.52	VTSS_EVCS	589

8.5.2.53	VTSS_ISDX_NONE	589
8.5.2.54	VTSS_AGGRS	589
8.5.2.55	VTSS_AGGR_NO_NONE	590
8.5.2.56	VTSS_AGGR_NO_START	590
8.5.2.57	VTSS_AGGR_NO_END	590
8.5.2.58	VTSS_GLAGS	590
8.5.2.59	VTSS_GLAG_NO_NONE	590
8.5.2.60	VTSS_GLAG_NO_START	591
8.5.2.61	VTSS_GLAG_NO_END	591
8.5.2.62	VTSS_GLAG_PORTS	591
8.5.2.63	VTSS_GLAG_PORT_START	591
8.5.2.64	VTSS_GLAG_PORT_END	591
8.5.2.65	VTSS_GLAG_PORT_ARRAY_SIZE	592
8.5.2.66	VTSS_PACKET_RX_QUEUE_CNT	592
8.5.2.67	VTSS_PACKET_RX_GRP_CNT	592
8.5.2.68	VTSS_PACKET_TX_GRP_CNT	592
8.5.2.69	VTSS_PACKET_RX_QUEUE_NONE	592
8.5.2.70	VTSS_PACKET_RX_QUEUE_START	593
8.5.2.71	VTSS_PACKET_RX_QUEUE_END	593
8.5.2.72	VTSS_ACL_POLICERS	593
8.5.2.73	VTSS_ACL_POLICER_NO_START	593
8.5.2.74	VTSS_ACL_POLICER_NO_END	593
8.5.2.75	VTSS_ACL_POLICY_NO_NONE	594
8.5.2.76	VTSS_ACL_POLICY_NO_MIN	594
8.5.2.77	VTSS_ACL_POLICY_NO_MAX	594
8.5.2.78	VTSS_ACL_POLICIES	594
8.5.2.79	VTSS_ACL_POLICY_NO_START	594
8.5.2.80	VTSS_ACL_POLICY_NO_END	595
8.5.2.81	VTSS_HQOS_COUNT	595
8.5.2.82	VTSS_HQOS_ID_NONE	595

8.5.2.83	VTSS_ONE_MIA	595
8.5.2.84	VTSS_ONE_MILL	595
8.5.2.85	VTSS_MAX_TIMEINTERVAL	596
8.5.2.86	VTSS_INTERVAL_SEC	596
8.5.2.87	VTSS_INTERVAL_MS	596
8.5.2.88	VTSS_INTERVAL_US	596
8.5.2.89	VTSS_INTERVAL_NS	596
8.5.2.90	VTSS_INTERVAL_PS	597
8.5.2.91	VTSS_SEC_NS_INTERVAL	597
8.5.2.92	VTSS_CLOCK_IDENTITY_LENGTH	597
8.5.2.93	VTSS_SYNCE_CLK_PORT_ARRAY_SIZE	597
8.5.3	Typedef Documentation	597
8.5.3.1	i8	598
8.5.3.2	i16	598
8.5.3.3	i32	598
8.5.3.4	i64	598
8.5.3.5	u8	598
8.5.3.6	u16	599
8.5.3.7	u32	599
8.5.3.8	u64	599
8.5.3.9	BOOL	599
8.5.3.10	uintptr_t	599
8.5.3.11	vtss_mac_addr_t	600
8.5.3.12	vtss_isdx_t	600
8.5.3.13	vtss_packet_rx_grp_t	600
8.5.3.14	vtss_packet_tx_grp_t	600
8.5.4	Enumeration Type Documentation	600
8.5.4.1	anonymous enum	600
8.5.4.2	vtss_mem_flags_t	603
8.5.4.3	vtss_port_interface_t	603

8.5.4.4	vtss_serdes_mode_t	604
8.5.4.5	vtss_storm_policer_mode_t	605
8.5.4.6	vtss_policer_type_t	605
8.5.4.7	vtss_vlan_frame_t	605
8.5.4.8	vtss_packet_reg_type_t	606
8.5.4.9	vtss_vdd_t	606
8.5.4.10	vtss_ip_type_t	606
8.5.4.11	vtss_vcap_bit_t	607
8.5.4.12	vtss_vcap_vr_type_t	607
8.5.4.13	vtss_vcap_key_type_t	607
8.5.4.14	vtss_ece_dir_t	608
8.5.4.15	vtss_ece_pop_tag_t	608
8.5.4.16	vtss_ece_inner_tag_type_t	608
8.5.4.17	vtss_hqos_sch_mode_t	609
8.6	vtss_api/include/vtss_ae_api.h File Reference	609
8.6.1	Detailed Description	609
8.7	vtss_api/include/vtss_afi_api.h File Reference	609
8.7.1	Detailed Description	609
8.8	vtss_api/include/vtss_aneg_api.h File Reference	610
8.8.1	Detailed Description	610
8.9	vtss_api/include/vtss_api.h File Reference	610
8.9.1	Detailed Description	610
8.10	vtss_api/include/vtss_evc_api.h File Reference	610
8.10.1	Detailed Description	612
8.10.2	Macro Definition Documentation	612
8.10.2.1	VTSS_EVC_POLICERS	613
8.10.2.2	VTSS_EVC_POLICER_ID_DISCARD	613
8.10.2.3	VTSS_EVC_POLICER_ID_NONE	613
8.10.2.4	VTSS_EVC_POLICER_ID_EVC	613
8.10.2.5	VTSS_EVC_ID_NONE	613

8.10.2.6 VTSS_ECE_ID_LAST	614
8.10.3 Enumeration Type Documentation	614
8.10.3.1 vtss_ece_type_t	614
8.10.3.2 vtss_ece_port_t	614
8.10.4 Function Documentation	614
8.10.4.1 vtss_evc_port_conf_get()	615
8.10.4.2 vtss_evc_port_conf_set()	615
8.10.4.3 vtss_evc_policer_conf_get()	615
8.10.4.4 vtss_evc_policer_conf_set()	616
8.10.4.5 vtss_evc_add()	616
8.10.4.6 vtss_evc_del()	617
8.10.4.7 vtss_evc_get()	617
8.10.4.8 vtss_ece_init()	618
8.10.4.9 vtss_ece_add()	618
8.10.4.10 vtss_ece_del()	618
8.10.4.11 vtss_evc_counters_get()	619
8.10.4.12 vtss_evc_counters_clear()	619
8.10.4.13 vtss_ece_counters_get()	620
8.10.4.14 vtss_ece_counters_clear()	620
8.11 vtss_api/include/vtss_fdma_api.h File Reference	621
8.11.1 Detailed Description	621
8.12 vtss_api/include/vtss_gfp_api.h File Reference	621
8.12.1 Detailed Description	621
8.13 vtss_api/include/vtss_hqos_api.h File Reference	621
8.13.1 Detailed Description	621
8.14 vtss_api/include/vtss_i2c_api.h File Reference	622
8.14.1 Detailed Description	622
8.15 vtss_api/include/vtss_init_api.h File Reference	622
8.15.1 Detailed Description	624
8.15.2 Macro Definition Documentation	625

8.15.2.1	VTSS_I2C_NO_MULTIPLEXER	625
8.15.2.2	VTSS_QS_CONF_MAX	625
8.15.2.3	VTSS_QS_CONF_MIN	625
8.15.3	Typedef Documentation	625
8.15.3.1	vtss_reg_read_t	625
8.15.3.2	vtss_reg_write_t	626
8.15.3.3	vtss_i2c_read_t	626
8.15.3.4	vtss_i2c_write_t	627
8.15.3.5	vtss_spi_read_write_t	627
8.15.3.6	vtss_spi_32bit_read_write_t	628
8.15.3.7	vtss_spi_64bit_read_write_t	628
8.15.3.8	vtss_miim_read_t	629
8.15.3.9	vtss_miim_write_t	629
8.15.3.10	vtss_mmd_read_t	629
8.15.3.11	vtss_mmd_read_inc_t	630
8.15.3.12	vtss_mmd_write_t	630
8.15.4	Enumeration Type Documentation	631
8.15.4.1	vtss_target_type_t	631
8.15.4.2	vtss_port_mux_mode_t	632
8.15.4.3	vtss_qs_mode_t	632
8.15.4.4	vtss_restart_t	633
8.15.5	Function Documentation	633
8.15.5.1	vtss_inst_get()	633
8.15.5.2	vtss_inst_create()	633
8.15.5.3	vtss_inst_destroy()	634
8.15.5.4	vtss_init_conf_get()	634
8.15.5.5	vtss_init_conf_set()	635
8.15.5.6	vtss_restart_conf_end()	635
8.15.5.7	vtss_restart_status_get()	635
8.15.5.8	vtss_restart_conf_get()	636

8.15.5.9 vtss_restart_conf_set()	636
8.15.5.10 vtss_qs_conf_set()	636
8.15.5.11 vtss_qs_conf_get()	637
8.16 vtss_api/include/vtss_l2_api.h File Reference	637
8.16.1 Detailed Description	645
8.16.2 Macro Definition Documentation	646
8.16.2.1 VTSS_MAC_ADDRS	646
8.16.2.2 VTSS_VSTAX_UPSIDS	646
8.16.2.3 VTSS_VSTAX_UPSID_START	646
8.16.2.4 VTSS_VSTAX_UPSID_MIN	646
8.16.2.5 VTSS_VSTAX_UPSID_MAX	646
8.16.2.6 VTSS_VSTAX_UPSID_LEGAL	647
8.16.2.7 VTSS_VSTAX_UPSID_UNDEF	647
8.16.2.8 VTSS_UPSPN_CPU	647
8.16.2.9 VTSS_UPSPN_NONE	647
8.16.2.10 VTSS_MSTIS	647
8.16.2.11 VTSS_MSTI_START	648
8.16.2.12 VTSS_MSTI_END	648
8.16.2.13 VTSS_MSTI_ARRAY_SIZE	648
8.16.2.14 VTSS_VCL_IDS	648
8.16.2.15 VTSS_VCL_ID_START	648
8.16.2.16 VTSS_VCL_ID_END	649
8.16.2.17 VTSS_VCL_ARRAY_SIZE	649
8.16.2.18 VTSS_VCE_ID_LAST	649
8.16.2.19 VTSS_VLAN_TRANS_GROUP_MAX_CNT	649
8.16.2.20 VTSS_VLAN_TRANS_MAX_CNT	649
8.16.2.21 VTSS_VLAN_TRANS_NULL_GROUP_ID	650
8.16.2.22 VTSS_VLAN_TRANS_FIRST_GROUP_ID	650
8.16.2.23 VTSS_VLAN_TRANS_VID_START	650
8.16.2.24 VTSS_VLAN_TRANS_MAX_VLAN_ID	650

8.16.2.25 VTSS_VLAN_TRANS_LAST_GROUP_ID	650
8.16.2.26 VTSS_VLAN_TRANS_VALID_GROUP_CHECK	651
8.16.2.27 VTSS_VLAN_TRANS_VALID_VLAN_CHECK	651
8.16.2.28 VTSS_VLAN_TRANS_NULL_CHECK	651
8.16.2.29 VTSS_VLAN_TRANS_PORT_BF_SIZE	651
8.16.2.30 VTSS_PVLANS	652
8.16.2.31 VTSS_PVLAN_NO_START	652
8.16.2.32 VTSS_PVLAN_NO_END	652
8.16.2.33 VTSS_PVLAN_ARRAY_SIZE	652
8.16.2.34 VTSS_PVLAN_NO_DEFAULT	652
8.16.2.35 VTSS_ERPIS	653
8.16.2.36 VTSS_ERPI_START	653
8.16.2.37 VTSS_ERPI_END	653
8.16.2.38 VTSS_ERPI_ARRAY_SIZE	653
8.16.3 Typedef Documentation	653
8.16.3.1 vtss_vt_id_t	653
8.16.4 Enumeration Type Documentation	653
8.16.4.1 vtss_stp_state_t	653
8.16.4.2 vtss_vlan_port_type_t	654
8.16.4.3 vtss_vlan_tx_tag_t	654
8.16.4.4 vtss_vce_type_t	654
8.16.4.5 vtss_mirror_tag_t	655
8.16.4.6 vtss_eps_port_type_t	655
8.16.4.7 vtss_eps_selector_t	655
8.16.4.8 vtss_erps_state_t	656
8.16.4.9 vtss_vstax_topology_type_t	656
8.16.5 Function Documentation	656
8.16.5.1 vtss_mac_table_add()	656
8.16.5.2 vtss_mac_table_del()	658
8.16.5.3 vtss_mac_table_get()	658

8.16.5.4 vtss_mac_table_get_next()	659
8.16.5.5 vtss_mac_table_age_time_get()	659
8.16.5.6 vtss_mac_table_age_time_set()	659
8.16.5.7 vtss_mac_table_age()	660
8.16.5.8 vtss_mac_table_vlan_age()	660
8.16.5.9 vtss_mac_table_flush()	661
8.16.5.10 vtss_mac_table_port_flush()	661
8.16.5.11 vtss_mac_table_vlan_flush()	661
8.16.5.12 vtss_mac_table_vlan_port_flush()	662
8.16.5.13 vtss_mac_table_upsid_flush()	662
8.16.5.14 vtss_mac_table_upsid_upspn_flush()	662
8.16.5.15 vtss_mac_table_glag_add()	663
8.16.5.16 vtss_mac_table_glag_flush()	663
8.16.5.17 vtss_mac_table_vlan_glag_flush()	664
8.16.5.18 vtss_mac_table_status_get()	664
8.16.5.19 vtss_learn_port_mode_get()	664
8.16.5.20 vtss_learn_port_mode_set()	665
8.16.5.21 vtss_port_state_get()	665
8.16.5.22 vtss_port_state_set()	666
8.16.5.23 vtss_stp_port_state_get()	666
8.16.5.24 vtss_stp_port_state_set()	667
8.16.5.25 vtss_mstp_vlan_msti_get()	667
8.16.5.26 vtss_mstp_vlan_msti_set()	667
8.16.5.27 vtss_mstp_port_msti_state_get()	668
8.16.5.28 vtss_mstp_port_msti_state_set()	668
8.16.5.29 vtss_vlan_conf_get()	669
8.16.5.30 vtss_vlan_conf_set()	669
8.16.5.31 vtss_vlan_port_conf_get()	669
8.16.5.32 vtss_vlan_port_conf_set()	671
8.16.5.33 vtss_vlan_port_members_get()	671

8.16.5.34 vtss_vlan_port_members_set()	672
8.16.5.35 vtss_vlan_vid_conf_get()	672
8.16.5.36 vtss_vlan_vid_conf_set()	673
8.16.5.37 vtss_vlan_tx_tag_get()	673
8.16.5.38 vtss_vlan_tx_tag_set()	673
8.16.5.39 vtss_vcl_port_conf_get()	674
8.16.5.40 vtss_vcl_port_conf_set()	674
8.16.5.41 vtss_vce_init()	675
8.16.5.42 vtss_vce_add()	675
8.16.5.43 vtss_vce_del()	676
8.16.5.44 vtss_vlan_trans_group_add()	676
8.16.5.45 vtss_vlan_trans_group_del()	676
8.16.5.46 vtss_vlan_trans_group_get()	677
8.16.5.47 vtss_vlan_trans_group_to_port_set()	677
8.16.5.48 vtss_vlan_trans_group_to_port_get()	678
8.16.5.49 vtss_isolated_vlan_get()	678
8.16.5.50 vtss_isolated_vlan_set()	679
8.16.5.51 vtss_isolated_port_members_get()	679
8.16.5.52 vtss_isolated_port_members_set()	679
8.16.5.53 vtss_pvlan_port_members_get()	680
8.16.5.54 vtss_pvlan_port_members_set()	680
8.16.5.55 vtss_apvlan_port_members_get()	681
8.16.5.56 vtss_apvlan_port_members_set()	681
8.16.5.57 vtss_dgroup_port_conf_get()	682
8.16.5.58 vtss_dgroup_port_conf_set()	682
8.16.5.59 vtss_sflow_port_conf_get()	682
8.16.5.60 vtss_sflow_port_conf_set()	683
8.16.5.61 vtss_sflow_sampling_rate_convert()	683
8.16.5.62 vtss_aggr_port_members_get()	684
8.16.5.63 vtss_aggr_port_members_set()	684

8.16.5.64 vtss_aggr_mode_get()	685
8.16.5.65 vtss_aggr_mode_set()	685
8.16.5.66 vtss_aggr_glag_members_get()	685
8.16.5.67 vtss_vstax_glag_get()	687
8.16.5.68 vtss_vstax_glag_set()	687
8.16.5.69 vtss_mirror_conf_get()	688
8.16.5.70 vtss_mirror_conf_set()	688
8.16.5.71 vtss_mirror_monitor_port_get()	688
8.16.5.72 vtss_mirror_monitor_port_set()	689
8.16.5.73 vtss_mirror_ingress_ports_get()	689
8.16.5.74 vtss_mirror_ingress_ports_set()	690
8.16.5.75 vtss_mirror_egress_ports_get()	690
8.16.5.76 vtss_mirror_egress_ports_set()	690
8.16.5.77 vtss_mirror_cpu_ingress_get()	691
8.16.5.78 vtss_mirror_cpu_ingress_set()	691
8.16.5.79 vtss_mirror_cpu_egress_get()	691
8.16.5.80 vtss_mirror_cpu_egress_set()	692
8.16.5.81 vtss_uc_flood_members_get()	692
8.16.5.82 vtss_uc_flood_members_set()	693
8.16.5.83 vtss_mc_flood_members_get()	693
8.16.5.84 vtss_mc_flood_members_set()	693
8.16.5.85 vtss_ipv4_mc_flood_members_get()	694
8.16.5.86 vtss_ipv4_mc_flood_members_set()	694
8.16.5.87 vtss_ipv6_mc_flood_members_get()	695
8.16.5.88 vtss_ipv6_mc_flood_members_set()	695
8.16.5.89 vtss_ipv6_mc_ctrl_flood_get()	695
8.16.5.90 vtss_ipv6_mc_ctrl_flood_set()	696
8.16.5.91 vtss_eps_port_conf_get()	696
8.16.5.92 vtss_eps_port_conf_set()	697
8.16.5.93 vtss_eps_port_selector_get()	697

8.16.5.94 vtss_eps_port_selector_set()	697
8.16.5.95 vtss_erps_vlan_member_get()	698
8.16.5.96 vtss_erps_vlan_member_set()	698
8.16.5.97 vtss_erps_port_state_get()	699
8.16.5.98 vtss_erps_port_state_set()	699
8.16.5.99 vtss_vstax_conf_get()	700
8.16.5.100 vtss_vstax_conf_set()	700
8.16.5.101 vtss_vstax_port_conf_get()	700
8.16.5.102 vtss_vstax_port_conf_set()	701
8.16.5.103 vtss_vstax_master_upsid_get()	701
8.16.5.104 vtss_vstax_master_upsid_set()	702
8.16.5.105 vtss_vstax_topology_set()	702
8.17 vtss_api/include/vtss_l3_api.h File Reference	702
8.17.1 Detailed Description	704
8.17.2 Macro Definition Documentation	704
8.17.2.1 VTSS_JR1_LPM_CNT	705
8.17.2.2 VTSS_LPM_CNT	705
8.17.2.3 VTSS_JR1_ARP_CNT	705
8.17.2.4 VTSS_ARP_CNT	705
8.17.2.5 VTSS_JR1_RLEG_CNT	705
8.17.2.6 VTSS_RLEG_CNT	706
8.17.2.7 VTSS_ARP_IPV4_RELATIONS	706
8.17.2.8 VTSS_ARP_IPV6_RELATIONS	706
8.17.3 Enumeration Type Documentation	706
8.17.3.1 vtss_l3_rleg_common_mode_t	706
8.17.3.2 vtss_l3_neighbour_type_t	707
8.17.4 Function Documentation	707
8.17.4.1 vtss_l3_flush()	707
8.17.4.2 vtss_l3_common_get()	707
8.17.4.3 vtss_l3_common_set()	708

8.17.4.4 vtss_l3_rleg_get()	708
8.17.4.5 vtss_l3_rleg_get_specific()	708
8.17.4.6 vtss_l3_rleg_addr()	709
8.17.4.7 vtss_l3_rleg_update()	709
8.17.4.8 vtss_l3_rleg_del()	710
8.17.4.9 vtss_l3_route_get()	710
8.17.4.10 vtss_l3_route_add()	710
8.17.4.11 vtss_l3_route_del()	711
8.17.4.12 vtss_l3_neighbour_get()	711
8.17.4.13 vtss_l3_neighbour_add()	712
8.17.4.14 vtss_l3_neighbour_del()	712
8.17.4.15 vtss_l3_counters_reset()	712
8.17.4.16 vtss_l3_counters_system_get()	713
8.17.4.17 vtss_l3_counters_rleg_get()	713
8.17.4.18 vtss_l3_counters_rleg_clear()	714
8.18 vtss_api/include/vtss_mac10g_api.h File Reference	714
8.18.1 Detailed Description	714
8.19 vtss_api/include/vtss_misc_api.h File Reference	714
8.19.1 Detailed Description	719
8.19.2 Macro Definition Documentation	720
8.19.2.1 VTSS_OS_TIMESTAMP_TYPE	720
8.19.2.2 VTSS_OS_TIMESTAMP	720
8.19.3 Typedef Documentation	720
8.19.3.1 tod_get_ns_cnt_cb_t	720
8.19.4 Enumeration Type Documentation	720
8.19.4.1 vtss_trace_layer_t	720
8.19.4.2 vtss_trace_group_t	721
8.19.4.3 vtss_trace_level_t	721
8.19.4.4 vtss_debug_layer_t	722
8.19.4.5 vtss_debug_group_t	722

8.19.4.6 <code>vtss_gpio_mode_t</code>	723
8.19.4.7 <code>vtss_sgpio_mode_t</code>	724
8.19.4.8 <code>vtss_sgpio_bmode_t</code>	724
8.19.4.9 <code>vtss_irq_t</code>	724
8.19.4.10 <code>vtss_eee_state_select_t</code>	725
8.19.5 Function Documentation	725
8.19.5.1 <code>vtss_trace_conf_get()</code>	725
8.19.5.2 <code>vtss_trace_conf_set()</code>	726
8.19.5.3 <code>vtss_callout_trace_printf()</code>	726
8.19.5.4 <code>vtss_callout_trace_hex_dump()</code>	727
8.19.5.5 <code>vtss_debug_info_get()</code>	727
8.19.5.6 <code>vtss_debug_info_print()</code>	728
8.19.5.7 <code>vtss_callout_lock()</code>	728
8.19.5.8 <code>vtss_callout_unlock()</code>	728
8.19.5.9 <code>vtss_debug_lock()</code>	729
8.19.5.10 <code>vtss_debug_unlock()</code>	729
8.19.5.11 <code>vtss_reg_read()</code>	729
8.19.5.12 <code>vtss_reg_write()</code>	730
8.19.5.13 <code>vtss_reg_write_masked()</code>	730
8.19.5.14 <code>vtss_intr_sticky_clear()</code>	731
8.19.5.15 <code>vtss_chip_id_get()</code>	731
8.19.5.16 <code>vtss_poll_1sec()</code>	732
8.19.5.17 <code>vtss_ptp_event_poll()</code>	732
8.19.5.18 <code>vtss_ptp_event_enable()</code>	733
8.19.5.19 <code>vtss_dev_all_event_poll()</code>	733
8.19.5.20 <code>vtss_dev_all_event_enable()</code>	734
8.19.5.21 <code>vtss_gpio_mode_set()</code>	734
8.19.5.22 <code>vtss_gpio_direction_set()</code>	734
8.19.5.23 <code>vtss_gpio_read()</code>	735
8.19.5.24 <code>vtss_gpio_write()</code>	735

8.19.5.25 vtss_gpio_event_poll()	736
8.19.5.26 vtss_gpio_event_enable()	736
8.19.5.27 vtss_sgpio_conf_get()	737
8.19.5.28 vtss_sgpio_conf_set()	737
8.19.5.29 vtss_sgpio_read()	738
8.19.5.30 vtss_sgpio_event_poll()	738
8.19.5.31 vtss_sgpio_event_enable()	739
8.19.5.32 vtss_intr_cfg()	739
8.19.5.33 vtss_irq_conf_get()	740
8.19.5.34 vtss_irq_conf_set()	740
8.19.5.35 vtss_irq_status_get_and_mask()	740
8.19.5.36 vtss_irq_enable()	741
8.19.5.37 vtss_tod_get_ns_cnt()	741
8.19.5.38 vtss_tod_set_ns_cnt_cb()	741
8.19.5.39 vtss_temp_sensor_init()	742
8.19.5.40 vtss_temp_sensor_get()	742
8.19.5.41 vtss_fan_rotation_get()	742
8.19.5.42 vtss_fan_cool_lvl_set()	743
8.19.5.43 vtss_fan_controller_init()	743
8.19.5.44 vtss_fan_cool_lvl_get()	744
8.19.5.45 vtss_eee_port_conf_set()	744
8.19.5.46 vtss_eee_port_state_set()	744
8.19.5.47 vtss_eee_port_counter_get()	745
8.19.5.48 vtss_debug_reg_check_set()	745
8.20 vtss_api/include/vtss_mpls_api.h File Reference	746
8.20.1 Detailed Description	746
8.21 vtss_api/include/vtss_oam_api.h File Reference	746
8.21.1 Detailed Description	747
8.22 vtss_api/include/vtss_oha_api.h File Reference	747
8.22.1 Detailed Description	747

8.23 vtss_api/include/vtss_os.h File Reference	747
8.23.1 Detailed Description	747
8.24 vtss_api/include/vtss_os_custom.h File Reference	747
8.24.1 Detailed Description	748
8.24.2 Macro Definition Documentation	748
8.24.2.1 uint	748
8.24.2.2 ulong	748
8.24.2.3 VTSS_MSLEEP	749
8.24.2.4 VTSS_MTIMER_START	749
8.24.2.5 VTSS_MTIMER_TIMEOUT	749
8.24.2.6 VTSS_MTIMER_CANCEL	749
8.24.2.7 VTSS_DIV64	749
8.24.2.8 VTSS_MOD64	750
8.24.2.9 VTSS_LABS	750
8.24.2.10 VTSS_LLabs	750
8.24.2.11 VTSS_OS_Ctz	750
8.24.2.12 VTSS_OS_Ctz64	751
8.24.2.13 VTSS_OS_MALLOC	751
8.24.2.14 VTSS_OS_FREE	751
8.24.2.15 VTSS_OS_RAND	751
8.24.3 Typedef Documentation	752
8.24.3.1 vtss_mtimer_t	752
8.25 vtss_api/include/vtss_os_ecos.h File Reference	752
8.25.1 Detailed Description	753
8.25.2 Macro Definition Documentation	753
8.25.2.1 VTSS_MSLEEP	753
8.25.2.2 VTSS_NSLEEP	754
8.25.2.3 VTSS_MTIMER_START	754
8.25.2.4 VTSS_MTIMER_TIMEOUT	754
8.25.2.5 VTSS_MTIMER_CANCEL	754

8.25.2.6 VTSS_TIME_OF_DAY	754
8.25.2.7 VTSS_DIV64	755
8.25.2.8 VTSS_MOD64	755
8.25.2.9 VTSS_LABS	755
8.25.2.10 VTSS_LLABS	755
8.25.2.11 VTSS_OS_CTZ	756
8.25.2.12 VTSS_OS_CTZ64	756
8.25.2.13 VTSS_OS_MALLOC	756
8.25.2.14 VTSS_OS_FREE	757
8.25.2.15 VTSS_OS_RAND	757
8.25.2.16 VTSS_OS_REORDER_BARRIER	757
8.25.2.17 VTSS_OS_COMPILER_ATTRIBUTE_ALIGNED	757
8.25.2.18 VTSS_OS_DCACHE_LINE_SIZE_BYTES	758
8.25.2.19 VTSS_OS_DCACHE_INVALIDATE	758
8.25.2.20 VTSS_OS_DCACHE_FLUSH	758
8.25.2.21 VTSS_OS_VIRT_TO_PHYS	758
8.25.2.22 VTSS_OS_BIG_ENDIAN	759
8.25.2.23 VTSS_OS_NTOHL	759
8.25.2.24 VTSS_OS_SCHEDULER_FLAGS	759
8.25.2.25 VTSS_OS_SCHEDULER_LOCK	759
8.25.2.26 VTSS_OS_SCHEDULER_UNLOCK	760
8.25.2.27 VTSS_OS_INTERRUPT_FLAGS	760
8.25.2.28 VTSS_OS_INTERRUPT_DISABLE	760
8.25.2.29 VTSS_OS_INTERRUPT_RESTORE	760
8.25.3 Typedef Documentation	760
8.25.3.1 vtss_mtimer_t	761
8.25.4 Function Documentation	761
8.25.4.1 llabs()	761
8.25.4.2 vtss_callout_malloc()	761
8.25.4.3 vtss_callout_free()	762

8.26 vtss_api/include/vtss_os_linux.h File Reference	762
8.26.1 Detailed Description	763
8.26.2 Macro Definition Documentation	763
8.26.2.1 VTSS_OS_BIG_ENDIAN	763
8.26.2.2 VTSS_OS_NTOHL	763
8.26.2.3 VTSS_NSLEEP	763
8.26.2.4 VTSS_MSLEEP	764
8.26.2.5 VTSS_MTIMER_START	764
8.26.2.6 VTSS_MTIMER_TIMEOUT	765
8.26.2.7 VTSS_MTIMER_CANCEL	765
8.26.2.8 VTSS_TIME_OF_DAY	765
8.26.2.9 VTSS_OS_SCHEDULER_FLAGS	765
8.26.2.10 VTSS_OS_SCHEDULER_LOCK	766
8.26.2.11 VTSS_OS_SCHEDULER_UNLOCK	766
8.26.2.12 VTSS_DIV64	766
8.26.2.13 VTSS_MOD64	766
8.26.2.14 VTSS_LABS	767
8.26.2.15 VTSS_LLabs	767
8.26.2.16 VTSS_OS_CTZ	767
8.26.2.17 VTSS_OS_CTZ64	768
8.26.2.18 VTSS_OS_MALLOC	768
8.26.2.19 VTSS_OS_FREE	768
8.26.2.20 VTSS_OS_RAND	769
8.27 vtss_api/include/vtss_otn_api.h File Reference	769
8.27.1 Detailed Description	769
8.28 vtss_api/include/vtss_packet_api.h File Reference	769
8.28.1 Detailed Description	773
8.28.2 Macro Definition Documentation	773
8.28.2.1 VTSS_PRIO_SUPER	773
8.28.2.2 VTSS_VSTAX_TTL_PORT	773

8.28.2.3 VTSS_VSTAX_HDR_SIZE	773
8.28.2.4 VTSS_JR1_PACKET_HDR_SIZE_BYTES	773
8.28.2.5 VTSS_JR2_PACKET_HDR_SIZE_BYTES	774
8.28.2.6 VTSS_SVL_PACKET_HDR_SIZE_BYTES	774
8.28.2.7 VTSS_L26_PACKET_HDR_SIZE_BYTES	774
8.28.2.8 VTSS_PACKET_HDR_SIZE_BYTES	774
8.28.2.9 VTSS_SVL_RX_IFH_SIZE	774
8.28.2.10 VTSS_JR1_RX_IFH_SIZE	775
8.28.2.11 VTSS_L26_RX_IFH_SIZE	775
8.28.2.12 VTSS_JR2_RX_IFH_SIZE	775
8.28.2.13 VTSS_PACKET_TX_IFH_MAX	775
8.28.3 Enumeration Type Documentation	775
8.28.3.1 vtss_packet_filter_t	775
8.28.3.2 vtss_vstax_fwd_mode_t	776
8.28.3.3 vtss_packet_oam_type_t	776
8.28.3.4 vtss_packet_ptp_action_t	777
8.28.3.5 vtss_tag_type_t	777
8.28.3.6 vtss_packet_rx_hints_t	777
8.28.3.7 vtss_packet_tx_vstax_t	778
8.28.4 Function Documentation	779
8.28.4.1 vtss_npi_conf_get()	779
8.28.4.2 vtss_npi_conf_set()	779
8.28.4.3 vtss_packet_rx_conf_get()	780
8.28.4.4 vtss_packet_rx_conf_set()	780
8.28.4.5 vtss_packet_rx_port_conf_get()	780
8.28.4.6 vtss_packet_rx_port_conf_set()	782
8.28.4.7 vtss_packet_rx_frame_get()	782
8.28.4.8 vtss_packet_rx_frame_get_raw()	783
8.28.4.9 vtss_packet_rx_frame_discard()	783
8.28.4.10 vtss_packet_tx_frame_port()	784

8.28.4.11 <code>vtss_packet_tx_frame_port_vlan()</code>	784
8.28.4.12 <code>vtss_packet_tx_frame_vlan()</code>	785
8.28.4.13 <code>vtss_packet_frame_info_init()</code>	785
8.28.4.14 <code>vtss_packet_frame_filter()</code>	785
8.28.4.15 <code>vtss_packet_port_info_init()</code>	786
8.28.4.16 <code>vtss_packet_port_filter_get()</code>	786
8.28.4.17 <code>vtss_packet_tx_frame_vstax()</code>	787
8.28.4.18 <code>vtss_packet_vstax_header2frame()</code>	787
8.28.4.19 <code>vtss_packet_vstax_frame2header()</code>	788
8.28.4.20 <code>vtss_packet_rx_hdr_decode()</code>	788
8.28.4.21 <code>vtss_packet_tx_hdr_encode()</code>	789
8.28.4.22 <code>vtss_packet_tx_hdr_compile()</code>	789
8.28.4.23 <code>vtss_packet_tx_frame()</code>	790
8.28.4.24 <code>vtss_packet_tx_info_init()</code>	790
8.28.4.25 <code>vtss_packet_dma_conf_get()</code>	791
8.28.4.26 <code>vtss_packet_dma_conf_set()</code>	791
8.28.4.27 <code>vtss_packet_dma_offset()</code>	792
8.29 <code>vtss_api/include/vtss_pcs_10gbase_r_api.h</code> File Reference	792
8.29.1 Detailed Description	792
8.30 <code>vtss_api/include/vtss_phy_10g_api.h</code> File Reference	792
8.30.1 Detailed Description	806
8.30.2 Macro Definition Documentation	806
8.30.2.1 <code>VTSS_SLEWRATE_25PS</code>	807
8.30.2.2 <code>VTSS_SLEWRATE_35PS</code>	807
8.30.2.3 <code>VTSS_SLEWRATE_55PS</code>	807
8.30.2.4 <code>VTSS_SLEWRATE_70PS</code>	807
8.30.2.5 <code>VTSS_SLEWRATE_120PS</code>	807
8.30.2.6 <code>VTSS_SLEWRATE_INVALID</code>	808
8.30.2.7 <code>BOOLEAN_STORAGE_COUNT</code>	808
8.30.2.8 <code>UNSIGNED_STORAGE_COUNT</code>	808

8.30.2.9 PHASE_POINTS	808
8.30.2.10 AMPLITUDE_POINTS	808
8.30.2.11 VTSS_PHY_10G_ONE_LINE_ACTIVE	809
8.30.2.12 VTSS_PHY_10G_MACSEC_DISABLED	809
8.30.2.13 VTSS_PHY_10G_TIMESTAMP_DISABLED	809
8.30.2.14 VTSS_PHY_10G_MACSEC_KEY_128	809
8.30.2.15 VTSS_10G_PHY_GPIO_MAX	809
8.30.2.16 VTSS_10G_PHY_GPIO_MAL_MAX	810
8.30.2.17 VTSS_PHY_10G_LINK_LOS_EV	810
8.30.2.18 VTSS_PHY_10G_RX_LOL_EV [1/2]	810
8.30.2.19 VTSS_PHY_10G_TX_LOL_EV [1/2]	810
8.30.2.20 VTSS_PHY_10G_LOPC_EV	810
8.30.2.21 VTSS_PHY_10G_HIGH_BER_EV	811
8.30.2.22 VTSS_PHY_10G_MODULE_STAT_EV	811
8.30.2.23 VTSS_PHY_10G_PCS_RECEIVE_FAULT_EV	811
8.30.2.24 VTSS_PHY_EWIS_SEF_EV	811
8.30.2.25 VTSS_PHY_EWIS_FPLM_EV	811
8.30.2.26 VTSS_PHY_EWIS_FAIS_EV	812
8.30.2.27 VTSS_PHY_EWIS_LOF_EV	812
8.30.2.28 VTSS_PHY_EWIS_RDIL_EV	812
8.30.2.29 VTSS_PHY_EWIS_AISL_EV	812
8.30.2.30 VTSS_PHY_EWIS_LCDP_EV	812
8.30.2.31 VTSS_PHY_EWIS_PLMP_EV	813
8.30.2.32 VTSS_PHY_EWIS_AISP_EV	813
8.30.2.33 VTSS_PHY_EWIS_LOPP_EV	813
8.30.2.34 VTSS_PHY_EWIS_UNEQP_EV	813
8.30.2.35 VTSS_PHY_EWIS_FEUNEQP_EV	813
8.30.2.36 VTSS_PHY_EWIS_FERDIP_EV	814
8.30.2.37 VTSS_PHY_EWIS_REIL_EV	814
8.30.2.38 VTSS_PHY_EWIS_REIP_EV	814

8.30.2.39 VTSS_PHY_EWIS_B1_NZ_EV	814
8.30.2.40 VTSS_PHY_EWIS_B2_NZ_EV	814
8.30.2.41 VTSS_PHY_EWIS_B3_NZ_EV	815
8.30.2.42 VTSS_PHY_EWIS_REIL_NZ_EV	815
8.30.2.43 VTSS_PHY_EWIS_REIP_NZ_EV	815
8.30.2.44 VTSS_PHY_EWIS_B1_THRESH_EV	815
8.30.2.45 VTSS_PHY_EWIS_B2_THRESH_EV	815
8.30.2.46 VTSS_PHY_EWIS_B3_THRESH_EV	816
8.30.2.47 VTSS_PHY_EWIS_REIL_THRESH_EV	816
8.30.2.48 VTSS_PHY_EWIS_REIP_THRESH_EV	816
8.30.2.49 VTSS_PHY_10G_RX_LOS_EV	816
8.30.2.50 VTSS_PHY_10G_RX_LOL_EV [2/2]	816
8.30.2.51 VTSS_PHY_10G_TX_LOL_EV [2/2]	817
8.30.2.52 VTSS_PHY_10G_RX_CHAR_DEC_CNT_THRESH_EV	817
8.30.2.53 VTSS_PHY_10G_TX_CHAR_ENC_CNT_THRESH_EV	817
8.30.2.54 VTSS_PHY_10G_RX_BLK_DEC_CNT_THRESH_EV	817
8.30.2.55 VTSS_PHY_10G_TX_BLK_ENC_CNT_THRESH_EV	817
8.30.2.56 VTSS_PHY_10G_RX_SEQ_CNT_THRESH_EV	818
8.30.2.57 VTSS_PHY_10G_TX_SEQ_CNT_THRESH_EV	818
8.30.2.58 VTSS_PHY_10G_FEC_UNFIXED_CNT_THRESH_EV	818
8.30.2.59 VTSS_PHY_10G_FEC_FIXED_CNT_THRESH_EV	818
8.30.2.60 VTSS_PHY_10G_HIGHBER_EV	818
8.30.2.61 VTSS_PHY_10G_RX_LINK_STAT_EV [1/2]	819
8.30.2.62 VTSS_PHY_10G_RX_LINK_STAT_EV [2/2]	819
8.30.2.63 VTSS_PHY_10G_GPIO_INT_AGG0_EV	819
8.30.2.64 VTSS_PHY_10G_GPIO_INT_AGG1_EV	819
8.30.2.65 VTSS_PHY_10G_GPIO_INT_AGG2_EV	819
8.30.2.66 VTSS_PHY_10G_GPIO_INT_AGG3_EV	820
8.30.2.67 VTSS_PHY_1G_LINE_AUTONEG_RESTART_EV	820
8.30.2.68 VTSS_PHY_1G_HOST_AUTONEG_RESTART_EV	820

8.30.2.69 VTSS_PHY_10G_LINE_MAC_LOCAL_FAULT_EV	820
8.30.2.70 VTSS_PHY_10G_HOST_MAC_LOCAL_FAULT_EV	820
8.30.2.71 VTSS_PHY_10G_LINE_MAC_REMOTE_FAULT_EV	821
8.30.2.72 VTSS_PHY_10G_HOST_MAC_REMOTE_FAULT_EV	821
8.30.3 Typedef Documentation	821
8.30.3.1 vtss_gpio_10g_no_t	821
8.30.3.2 ckout_sel_t	821
8.30.3.3 vtss_32_cntr_t	821
8.30.3.4 vtss_phy_10g_event_t	822
8.30.3.5 vtss_phy_10g_extnd_event_t	822
8.30.4 Enumeration Type Documentation	822
8.30.4.1 oper_mode_t	822
8.30.4.2 vtss_wrefclk_t	823
8.30.4.3 vtss_phy_interface_mode	823
8.30.4.4 vtss_recvrd_t	824
8.30.4.5 vtss_recvrdclk_cdr_div_t	824
8.30.4.6 vtss_srefclk_div_t	824
8.30.4.7 vtss_wref_clk_div_t	825
8.30.4.8 apc_ib_regulator_t	825
8.30.4.9 ddr_mode_t	825
8.30.4.10 clk_mstr_t	826
8.30.4.11 vtss_rptr_rate_t	826
8.30.4.12 vtss_phy_10g_media_t	827
8.30.4.13 vtss_phy_6g_link_partner_distance_t	827
8.30.4.14 vtss_phy_10g_ib_apc_op_mode_t	827
8.30.4.15 vtss_channel_t	828
8.30.4.16 vtss_recvrd_clkout_t	828
8.30.4.17 vtss_phy_10g_srefclk_freq_t	829
8.30.4.18 vtss_phy_10g_ckout_freq_t	829
8.30.4.19 vtss_ckout_data_sel_t	829

8.30.4.20 <code>vtss_phy_10g_squelch_src_t</code>	830
8.30.4.21 <code>vtss_phy_10g_clk_sel_t</code>	831
8.30.4.22 <code>vtss_phy_10g_recvrd_clk_sel_t</code>	832
8.30.4.23 <code>ckout_sel_</code>	832
8.30.4.24 <code>vtss_phy_10g_sckout_freq_t</code>	832
8.30.4.25 <code>vtss_phy_10g_rx_macro_t</code>	833
8.30.4.26 <code>vtss_phy_10g_tx_macro_t</code>	833
8.30.4.27 <code>vtss_phy_10g_clause_37_remote_fault_t</code>	833
8.30.4.28 <code>vtss_lb_type_t</code>	834
8.30.4.29 <code>vtss_phy_10g_power_t</code>	835
8.30.4.30 <code>vtss_phy_10g_failover_mode_t</code>	835
8.30.4.31 <code>vtss_phy_10g_auto_failover_event_t</code>	835
8.30.4.32 <code>vtss_phy_10g_auto_failover_filter_t</code>	836
8.30.4.33 <code>vtss_phy_10g_vscope_scan_t</code>	836
8.30.4.34 <code>vtss_phy_10g_pkt_mon_rst_t</code>	837
8.30.4.35 <code>vtss_10g_phy_gpio_t</code>	837
8.30.4.36 <code>vtss_gpio_10g_gpio_intr_sgnl_t</code>	838
8.30.4.37 <code>vtss_gpio_10g_chan_intrpt_t</code>	840
8.30.4.38 <code>vtss_gpio_10g_aggr_intrpt_t</code>	841
8.30.4.39 <code>vtss_gpio_10g_input_t</code>	841
8.30.5 Function Documentation	842
8.30.5.1 <code>vtss_phy_10g_mode_get()</code>	842
8.30.5.2 <code>vtss_phy_10g_init()</code>	842
8.30.5.3 <code>vtss_phy_10g_mode_set()</code>	844
8.30.5.4 <code>vtss_phy_10g_ib_conf_set()</code>	844
8.30.5.5 <code>vtss_phy_10g_ib_conf_get()</code>	845
8.30.5.6 <code>vtss_phy_10g_ib_status_get()</code>	845
8.30.5.7 <code>vtss_phy_10g_apc_conf_set()</code>	846
8.30.5.8 <code>vtss_phy_10g_apc_conf_get()</code>	846
8.30.5.9 <code>vtss_phy_10g_apc_status_get()</code>	847

CONTENTS	CXV
8.30.5.10 vtss_phy_10g_apc_restart()	847
8.30.5.11 vtss_phy_10g_jitter_conf_set()	848
8.30.5.12 vtss_phy_10g_jitter_conf_get()	848
8.30.5.13 vtss_phy_10g_jitter_status_get()	849
8.30.5.14 vtss_phy_10g_sync_clkout_get()	849
8.30.5.15 vtss_phy_10g_sync_clkout_set()	850
8.30.5.16 vtss_phy_10g_xfp_clkout_get()	850
8.30.5.17 vtss_phy_10g_xfp_clkout_set()	851
8.30.5.18 vtss_phy_10g_rxckout_get()	851
8.30.5.19 vtss_phy_10g_rxckout_set()	852
8.30.5.20 vtss_phy_10g_txckout_get()	852
8.30.5.21 vtss_phy_10g_txckout_set()	852
8.30.5.22 vtss_phy_10g_srefclk_conf_get()	853
8.30.5.23 vtss_phy_10g_srefclk_conf_set()	853
8.30.5.24 vtss_phy_10g_sckout_conf_set()	854
8.30.5.25 vtss_phy_10g_ckout_conf_set()	854
8.30.5.26 vtss_phy_10g_line_clk_conf_set()	855
8.30.5.27 vtss_phy_10g_host_clk_conf_set()	855
8.30.5.28 vtss_phy_10g_line_recvrd_clk_conf_set()	856
8.30.5.29 vtss_phy_10g_host_recvrd_clk_conf_set()	856
8.30.5.30 vtss_phy_10g_lane_sync_set()	857
8.30.5.31 vtss_phy_10g_debug_register_dump()	857
8.30.5.32 vtss_phy_10g_base_kr_train_aneg_get()	858
8.30.5.33 vtss_phy_10g_base_kr_train_aneg_set()	858
8.30.5.34 vtss_phy_10g_base_host_kr_train_aneg_set()	858
8.30.5.35 vtss_phy_10g_base_kr_conf_get()	859
8.30.5.36 vtss_phy_10g_base_kr_conf_set()	859
8.30.5.37 vtss_phy_10g_base_kr_host_conf_get()	860
8.30.5.38 vtss_phy_10g_base_kr_host_conf_set()	860
8.30.5.39 vtss_phy_10g_kr_status_get()	861

8.30.5.40 vtss_phy_10g_ob_status_get()	861
8.30.5.41 vtss_phy_10g_status_get()	862
8.30.5.42 vtss_phy_10g_serdes_status_get()	862
8.30.5.43 vtss_phy_10g_reset()	863
8.30.5.44 vtss_phy_10g_clause_37_status_get()	863
8.30.5.45 vtss_phy_10g_clause_37_control_get()	863
8.30.5.46 vtss_phy_10g_clause_37_control_set()	864
8.30.5.47 vtss_phy_10g_loopback_set()	864
8.30.5.48 vtss_phy_10g_loopback_get()	865
8.30.5.49 vtss_phy_10g_cnt_get()	865
8.30.5.50 vtss_phy_10g_power_get()	866
8.30.5.51 vtss_phy_10g_power_set()	866
8.30.5.52 vtss_phy_10G_is_valid()	867
8.30.5.53 vtss_phy_10g_failover_set()	867
8.30.5.54 vtss_phy_10g_failover_get()	867
8.30.5.55 vtss_phy_10g_auto_failover_set()	868
8.30.5.56 vtss_phy_10g_auto_failover_get()	868
8.30.5.57 vtss_phy_10g_vscope_conf_set()	869
8.30.5.58 vtss_phy_10g_vscope_conf_get()	869
8.30.5.59 vtss_phy_10g_vscope_scan_status_get()	870
8.30.5.60 vtss_phy_10g_pcs_prbs_gen_conf_set()	870
8.30.5.61 vtss_phy_10g_pcs_prbs_gen_conf_get()	871
8.30.5.62 vtss_phy_10g_pcs_prbs_mon_conf_set()	871
8.30.5.63 vtss_phy_10g_pcs_prbs_mon_conf_get()	872
8.30.5.64 vtss_phy_10g_pcs_prbs_mon_status_get()	872
8.30.5.65 vtss_phy_10g_prbs_gen_conf()	873
8.30.5.66 vtss_phy_10g_prbs_gen_conf_get()	873
8.30.5.67 vtss_phy_10g_prbs_mon_conf()	873
8.30.5.68 vtss_phy_10g_prbs_mon_conf_get()	874
8.30.5.69 vtss_phy_10g_prbs_mon_status_get()	874

8.30.5.70 vtss_phy_10g_pkt_gen_conf()	875
8.30.5.71 vtss_phy_10g_pkt_mon_conf()	875
8.30.5.72 vtss_phy_10g_pkt_mon_counters_get()	876
8.30.5.73 vtss_phy_10g_id_get()	876
8.30.5.74 vtss_phy_10g_gpio_mode_set()	877
8.30.5.75 vtss_phy_10g_gpio_mode_get()	877
8.30.5.76 vtss_phy_10g_gpio_read()	878
8.30.5.77 vtss_phy_10g_gpio_write()	878
8.30.5.78 vtss_phy_10g_event_enable_set()	879
8.30.5.79 vtss_phy_10g_event_enable_get()	879
8.30.5.80 vtss_phy_10g_extended_event_enable_get()	879
8.30.5.81 vtss_phy_10g_event_poll()	880
8.30.5.82 vtss_phy_10g_pcs_status_get()	880
8.30.5.83 vtss_phy_10g_extended_event_poll()	881
8.30.5.84 vtss_phy_10g_extended_event_enable_set()	881
8.30.5.85 vtss_phy_10g_poll_1sec()	882
8.30.5.86 vtss_phy_10g_edc_fw_status_get()	882
8.30.5.87 vtss_phy_10g_fc_buffer_reset()	882
8.30.5.88 vtss_phy_10g_csr_read()	883
8.30.5.89 vtss_phy_10g_csr_write()	883
8.30.5.90 vtss_phy_warm_start_10g_failed_get()	884
8.30.5.91 vtss_phy_10g_sgmii_mode_set()	884
8.30.5.92 vtss_phy_10g_i2c_read()	885
8.30.5.93 vtss_phy_10g_i2c_write()	885
8.30.5.94 vtss_phy_10g_get_user_data()	885
8.31 vtss_api/include/vtss_phy_api.h File Reference	886
8.31.1 Detailed Description	896
8.31.2 Macro Definition Documentation	896
8.31.2.1 MAX_CFG_BUF_SIZE	896
8.31.2.2 MAX_STAT_BUF_SIZE	896

8.31.2.3 VTSS_PHY_POWER_ACTIPHY_BIT	896
8.31.2.4 VTSS_PHY_POWER_DYNAMIC_BIT	897
8.31.2.5 VTSS_PHY_ACTIPHY_PWR	897
8.31.2.6 VTSS_PHY_LINK_DOWN_PWR	897
8.31.2.7 VTSS_PHY_LINK_UP_FULL_PWR	897
8.31.2.8 VTSS_PHY_RECov_CLK1	897
8.31.2.9 VTSS_PHY_RECov_CLK2	898
8.31.2.10 VTSS_PHY_RECov_CLK_NUM	898
8.31.2.11 VTSS_PHY_PAGE_STANDARD	898
8.31.2.12 VTSS_PHY_PAGE_EXTENDED	898
8.31.2.13 VTSS_PHY_PAGE_EXTENDED_2	898
8.31.2.14 VTSS_PHY_PAGE_EXTENDED_3	899
8.31.2.15 VTSS_PHY_PAGE_EXTENDED_4	899
8.31.2.16 VTSS_PHY_PAGE_GPIO	899
8.31.2.17 VTSS_PHY_PAGE_1588	899
8.31.2.18 VTSS_PHY_PAGE_MACSEC	899
8.31.2.19 VTSS_PHY_PAGE_TEST	900
8.31.2.20 VTSS_PHY_PAGE_TR	900
8.31.2.21 VTSS_PHY_PAGE_0x2DAF	900
8.31.2.22 VTSS_PHY_REG_STANDARD	900
8.31.2.23 VTSS_PHY_REG_EXTENDED	900
8.31.2.24 VTSS_PHY_REG_GPIO	901
8.31.2.25 VTSS_PHY_REG_TEST	901
8.31.2.26 VTSS_PHY_REG_TR	901
8.31.2.27 VTSS_PHY_LINK_LOS_EV	901
8.31.2.28 VTSS_PHY_LINK_FFAIL_EV	901
8.31.2.29 VTSS_PHY_LINK_AMS_EV	902
8.31.2.30 VTSS_PHY_LINK_SPEED_STATE_CHANGE_EV	902
8.31.2.31 VTSS_PHY_LINK_FDX_STATE_CHANGE_EV	902
8.31.2.32 VTSS_PHY_LINK_AUTO_NEG_ERROR_EV	902

8.31.2.33 VTSS_PHY_LINK_AUTO_NEG_COMPLETE_EV	902
8.31.2.34 VTSS_PHY_LINK_INLINE_POW_DEV_DETECT_EV	903
8.31.2.35 VTSS_PHY_LINK_SYMBOL_ERR_INT_EV	903
8.31.2.36 VTSS_PHY_LINK_TX_FIFO_OVERFLOW_INT_EV	903
8.31.2.37 VTSS_PHY_LINK_RX_FIFO_OVERFLOW_INT_EV	903
8.31.2.38 VTSS_PHY_LINK_FALSE_CARRIER_INT_EV	903
8.31.2.39 VTSS_PHY_LINK_LINK_SPEED_DS_DETECT_EV	904
8.31.2.40 VTSS_PHY_LINK_MASTER_SLAVE_RES_ERR_EV	904
8.31.2.41 VTSS_PHY_LINK_RX_ER_INT_EV	904
8.31.2.42 VTSS_PHY_LINK_EXTENDED_REG_INT_EV	904
8.31.2.43 VTSS_PHY_LINK_WAKE_ON_LAN_INT_EV	904
8.31.2.44 VTSS_PHY_LINK_EXT_EEE_WAKE_ERR_EV	905
8.31.2.45 VTSS_PHY_LINK_EXT_EEE_WAIT_TS_EV	905
8.31.2.46 VTSS_PHY_LINK_EXT_EEE_WAIT_RX_TQ_EV	905
8.31.2.47 VTSS_PHY_LINK_EXT_EEE_LINKFAIL_EV	905
8.31.2.48 VTSS_PHY_LINK_EXT_RR_SW_COMPL_EV	905
8.31.2.49 VTSS_PHY_LINK_EXT_MACSEC_HOST_MAC_EV	906
8.31.2.50 VTSS_PHY_LINK_EXT_MACSEC_LINE_MAC_EV	906
8.31.2.51 VTSS_PHY_LINK_EXT_MACSEC_FC_BUFF_EV	906
8.31.2.52 VTSS_PHY_LINK_EXT_MACSEC_INGRESS_EV	906
8.31.2.53 VTSS_PHY_LINK_EXT_MACSEC_EGRESS_EV	906
8.31.2.54 VTSS_PHY_LINK_EXT_MEM_INT_RING_EV	907
8.31.2.55 MAX_WOL_MAC_ADDR_SIZE	907
8.31.2.56 MAX_WOL_PASSWD_SIZE	907
8.31.2.57 MIN_WOL_PASSWD_SIZE	907
8.31.3 Typedef Documentation	907
8.31.3.1 vtss_phy_recov_clk_t	907
8.31.3.2 vtss_phy_led_intensity	908
8.31.4 Enumeration Type Documentation	908
8.31.4.1 vtss_phy_led_mode_t	908

8.31.4.2 vtss_phy_media_interface_t	909
8.31.4.3 vtss_phy_mdi_t	909
8.31.4.4 rgmii_skew_delay_psec_t	910
8.31.4.5 vtss_phy_forced_reset_t	910
8.31.4.6 vtss_phy_pkt_mode_t	911
8.31.4.7 vtss_phy_mode_t	911
8.31.4.8 vtss_phy_fast_link_fail_t	911
8.31.4.9 vtss_phy_sigdet_polarity_t	912
8.31.4.10 vtss_phy_unidirectional_t	912
8.31.4.11 vtss_phy_mac_serdes_pcs_sgmii_pre	912
8.31.4.12 vtss_phy_media_rem_fault_t	913
8.31.4.13 vtss_phy_media_force_ams_sel_t	913
8.31.4.14 vtss_phy_clk_source_t	913
8.31.4.15 vtss_phy_freq_t	914
8.31.4.16 vtss_phy_clk_squelch	914
8.31.4.17 vtss_phy_gpio_mode_t	914
8.31.4.18 lb_type	915
8.31.4.19 vtss_wol_passwd_len_type_t	915
8.31.4.20 vtss_fefi_mode_t	916
8.31.5 Function Documentation	916
8.31.5.1 vtss_phy_pre_reset()	916
8.31.5.2 vtss_phy_post_reset()	916
8.31.5.3 vtss_phy_pre_system_reset()	917
8.31.5.4 vtss_phy_reset()	917
8.31.5.5 vtss_phy_reset_get()	918
8.31.5.6 vtss_phy_chip_temp_get()	918
8.31.5.7 vtss_phy_chip_temp_init()	918
8.31.5.8 vtss_phy_conf_get()	920
8.31.5.9 vtss_phy_conf_set()	920
8.31.5.10 vtss_phy_status_get()	921

8.31.5.11 vtss_phy_cl37_lp_abil_get()	921
8.31.5.12 vtss_phy_id_get()	922
8.31.5.13 vtss_phy_conf_1g_get()	922
8.31.5.14 vtss_phy_conf_1g_set()	922
8.31.5.15 vtss_phy_status_1g_get()	923
8.31.5.16 vtss_phy_power_conf_get()	923
8.31.5.17 vtss_phy_power_conf_set()	924
8.31.5.18 vtss_phy_power_status_get()	924
8.31.5.19 vtss_phy_clock_conf_set()	925
8.31.5.20 vtss_phy_clock_conf_get()	925
8.31.5.21 vtss_phy_i2c_read()	926
8.31.5.22 vtss_phy_i2c_write()	926
8.31.5.23 vtss_phy_read()	927
8.31.5.24 vtss_phy_read_page()	927
8.31.5.25 vtss_phy_mmd_read()	928
8.31.5.26 vtss_phy_mmd_write()	928
8.31.5.27 vtss_phy_write()	929
8.31.5.28 vtss_phy_write_masked()	929
8.31.5.29 vtss_phy_write_masked_page()	930
8.31.5.30 vtss_phy_gpio_mode()	930
8.31.5.31 vtss_phy_gpio_get()	931
8.31.5.32 vtss_phy_gpio_set()	931
8.31.5.33 vtss_phy_veriphy_start()	932
8.31.5.34 vtss_phy_veriphy_get()	932
8.31.5.35 vtss_phy_led_mode_set()	933
8.31.5.36 vtss_phy_led_intensity_set()	933
8.31.5.37 vtss_phy_led_intensity_get()	933
8.31.5.38 vtss_phy_enhanced_led_control_init()	934
8.31.5.39 vtss_phy_enhanced_led_control_init_get()	934
8.31.5.40 vtss_phy_coma_mode_disable()	935

8.31.5.41 vtss_phy_coma_mode_enable()	935
8.31.5.42 vga_adc_debug()	935
8.31.5.43 vtss_phy_port_eee_capable()	936
8.31.5.44 vtss_phy_eee_ena()	936
8.31.5.45 vtss_phy_eee_conf_get()	937
8.31.5.46 vtss_phy_eee_conf_set()	937
8.31.5.47 vtss_phy_eee_power_save_state_get()	938
8.31.5.48 vtss_phy_eee_link_partner_advertisements_get()	938
8.31.5.49 vtss_phy_event_enable_set()	938
8.31.5.50 vtss_phy_event_enable_get()	939
8.31.5.51 vtss_phy_event_poll()	939
8.31.5.52 vtss_squelch_workaround()	940
8.31.5.53 vtss_phy_csr_wr()	940
8.31.5.54 vtss_phy_csr_rd()	941
8.31.5.55 vtss_phy_statistic_get()	941
8.31.5.56 vtss_phy_do_page_chk_set()	942
8.31.5.57 vtss_phy_do_page_chk_get()	942
8.31.5.58 vtss_phy_loopback_set()	943
8.31.5.59 vtss_phy_loopback_get()	943
8.31.5.60 vtss_phy_is_8051_crc_ok()	944
8.31.5.61 vtss_phy_cfg_ob_post0()	944
8.31.5.62 vtss_phy_cfg_ib_cterm()	944
8.31.5.63 vtss_phy_atom12_patch_settings_get()	945
8.31.5.64 vtss_phy_reg_decode_status()	945
8.31.5.65 vtss_phy_flowcontrol_decode_status()	946
8.31.5.66 vtss_phy_debug_stat_print()	946
8.31.5.67 vtss_phy_warm_start_failed_get()	947
8.31.5.68 vtss_phy_debug_phyinfo_print()	947
8.31.5.69 vtss_phy_debug_register_dump()	948
8.31.5.70 vtss_phy_detect_base_ports()	948

8.31.5.71 vtss_phy_ext_connector_loopback()	949
8.31.5.72 vtss_phy_serdes_sgmii_loopback()	949
8.31.5.73 vtss_phy_serdes_fmedia_loopback()	950
8.31.5.74 vtss_phy_debug_regdump_print()	950
8.31.5.75 vtss_phy_wol_enable()	951
8.31.5.76 vtss_phy_wol_conf_get()	951
8.31.5.77 vtss_phy_wol_conf_set()	951
8.31.5.78 vtss_phy_patch_settings_get()	952
8.31.5.79 vtss_phy_serdes6g_rcpll_status_get()	952
8.31.5.80 vtss_phy_serdes1g_rcpll_status_get()	953
8.31.5.81 vtss_phy_lcpll_status_get()	953
8.31.5.82 vtss_phy_reset_lcpll()	954
8.31.5.83 vtss_phy_sd6g_ob_post_rd()	954
8.31.5.84 vtss_phy_sd6g_ob_post_wr()	955
8.31.5.85 vtss_phy_sd6g_ob_lev_rd()	955
8.31.5.86 vtss_phy_sd6g_ob_lev_wr()	956
8.31.5.87 vtss_phy_mac_media_inhibit_odd_start()	956
8.31.5.88 vtss_phy_fefi_get()	957
8.31.5.89 vtss_phy_fefi_set()	957
8.31.5.90 vtss_phy_fefi_detect()	957
8.31.5.91 vtss_phy_mse_100m_get()	958
8.31.5.92 vtss_phy_mse_1000m_get()	958
8.31.5.93 vtss_phy_read_tr_addr()	959
8.31.5.94 vtss_phy_is_viper_revB()	959
8.31.5.95 vtss_phy_ext_event_poll()	960
8.31.5.96 vtss_phy_status_inst_poll()	960
8.31.5.97 vtss_phy_macsec_csr_sd6g_rd()	961
8.31.5.98 vtss_phy_macsec_csr_sd6g_wr()	961
8.32 vtss_api/include/vtss_phy_ts_api.h File Reference	962
8.32.1 Detailed Description	971

8.32.2 Macro Definition Documentation	971
8.32.2.1 VTSS_PHY_TS_FIFO_SIG_SRC_IP	971
8.32.2.2 VTSS_PHY_TS_FIFO_SIG_DEST_IP	971
8.32.2.3 VTSS_PHY_TS_FIFO_SIG_MSG_TYPE	972
8.32.2.4 VTSS_PHY_TS_FIFO_SIG_DOMAIN_NUM	972
8.32.2.5 VTSS_PHY_TS_FIFO_SIG_SOURCE_PORT_ID	972
8.32.2.6 VTSS_PHY_TS_FIFO_SIG_SEQ_ID	972
8.32.2.7 VTSS_PHY_TS_FIFO_SIG_DEST_MAC	972
8.32.2.8 VTSS_PHY_TS_SIG_LEN	973
8.32.2.9 VTSS_PHY_TS_SIG_TIME_STAMP_LEN	973
8.32.2.10 VTSS_PHY_TS_SIG_DOMAIN_NUM_LEN	973
8.32.2.11 VTSS_PHY_TS_SIG_SEQ_ID_LEN	973
8.32.2.12 VTSS_PHY_TS_SIG_MSG_TYPE_LEN	973
8.32.2.13 VTSS_PHY_TS_SIG_SOURCE_PORT_ID_LEN	974
8.32.2.14 VTSS_PHY_TS_SIG_SEQUENCE_ID_LEN	974
8.32.2.15 VTSS_PHY_TS_SIG_DEST_IP_LEN	974
8.32.2.16 VTSS_PHY_TS_SIG_SRC_IP_LEN	974
8.32.2.17 VTSS_PHY_TS_SIG_DEST_MAC_LEN	974
8.32.2.18 VTSS_PTP_IP_1588_VERSION_2_1	975
8.32.2.19 VTSS_PHY_TS_ETH_ADDR_MATCH_48BIT	975
8.32.2.20 VTSS_PHY_TS_ETH_ADDR_MATCH_ANY_UNICAST	975
8.32.2.21 VTSS_PHY_TS_ETH_ADDR_MATCH_ANY_MULTICAST	975
8.32.2.22 VTSS_PHY_TS_ETH_MATCH_DEST_ADDR	975
8.32.2.23 VTSS_PHY_TS_ETH_MATCH_SRC_ADDR	976
8.32.2.24 VTSS_PHY_TS_ETH_MATCH_SRC_OR_DEST	976
8.32.2.25 VTSS_PHY_TS_TAG_TYPE_C	976
8.32.2.26 VTSS_PHY_TS_TAG_TYPE_S	976
8.32.2.27 VTSS_PHY_TS_TAG_TYPE_I	976
8.32.2.28 VTSS_PHY_TS_TAG_TYPE_B	977
8.32.2.29 VTSS_PHY_TS_TAG_RANGE_NONE	977

8.32.2.30 VTSS_PHY_TS_TAG_RANGE_OUTER	977
8.32.2.31 VTSS_PHY_TS_TAG_RANGE_INNER	977
8.32.2.32 VTSS_PHY_TS_IP_VER_4	977
8.32.2.33 VTSS_PHY_TS_IP_VER_6	978
8.32.2.34 VTSS_PHY_TS_IP_MATCH_SRC	978
8.32.2.35 VTSS_PHY_TS_IP_MATCH_DEST	978
8.32.2.36 VTSS_PHY_TS_IP_MATCH_SRC_OR_DEST	978
8.32.2.37 VTSS_PHY_TS_MPLS_STACK_DEPTH_1	978
8.32.2.38 VTSS_PHY_TS_MPLS_STACK_DEPTH_2	979
8.32.2.39 VTSS_PHY_TS_MPLS_STACK_DEPTH_3	979
8.32.2.40 VTSS_PHY_TS_MPLS_STACK_DEPTH_4	979
8.32.2.41 VTSS_PHY_TS_MPLS_STACK_REF_POINT_TOP	979
8.32.2.42 VTSS_PHY_TS_MPLS_STACK_REF_POINT_END	979
8.32.2.43 VTSS_PHY_TS_ENG_FLOW_VALID_FOR_CH0	980
8.32.2.44 VTSS_PHY_TS_ENG_FLOW_VALID_FOR_CH1	980
8.32.2.45 VTSS_PHY_TS_INGR_ENGINE_ERR	980
8.32.2.46 VTSS_PHY_TS_INGR_RW_PREAM_ERR	980
8.32.2.47 VTSS_PHY_TS_INGR_RW_FCS_ERR	981
8.32.2.48 VTSS_PHY_TS_EGR_ENGINE_ERR	981
8.32.2.49 VTSS_PHY_TS_EGR_RW_FCS_ERR	981
8.32.2.50 VTSS_PHY_TS_EGR_TIMESTAMP_CAPTURED	981
8.32.2.51 VTSS_PHY_TS_EGR_FIFO_OVERFLOW	981
8.32.2.52 VTSS_PHY_TS_DATA_IN_RSRVD_FIELD	982
8.32.2.53 VTSS_PHY_TS_LTC_NEW_PPS_INTRPT	982
8.32.2.54 VTSS_PHY_TS_LTC_LOAD_SAVE_NEW_TOD	982
8.32.2.55 VTSS_PHY_TS_8487_XAUI_SEL_0	982
8.32.2.56 VTSS_PHY_TS_8487_XAUI_SEL_1	982
8.32.2.57 VTSS_PHY_TS_INGR_DATAPATH_RESET	983
8.32.2.58 VTSS_PHY_TS_EGR_DATAPATH_RESET	983
8.32.2.59 VTSS_PHY_TS_INGR_LTC1_RESET	983

8.32.2.60 VTSS_PHY_TS_EGR_LTC2_RESET	983
8.32.2.61 VTSS_PHY_TS_EGR_FIFO_RESET	983
8.32.3 Typedef Documentation	984
8.32.3.1 vtss_phy_ts_alt_clock_mode_t	984
8.32.3.2 vtss_phy_ts_scaled_ppb_t	984
8.32.3.3 vtss_phy_ts_fifo_sig_mask_t	984
8.32.3.4 vtss_phy_ts_fifo_read	984
8.32.3.5 vtss_phy_ts_engine_channel_map_t	985
8.32.3.6 vtss_phy_ts_event_t	985
8.32.3.7 vtss_phy_ts_8487_xau1_sel_t	985
8.32.3.8 vtss_phy_ts_soft_reset_t	985
8.32.4 Enumeration Type Documentation	985
8.32.4.1 vtss_phy_ts_todadj_status_t	985
8.32.4.2 vtss_phy_ts_fifo_status_t	986
8.32.4.3 vtss_phy_ts_encap_t	986
8.32.4.4 vtss_phy_ts_engine_t	986
8.32.4.5 vtss_phy_ts_engine_flow_match_t	987
8.32.4.6 vtss_phy_ts_ptp_clock_mode_t	988
8.32.4.7 vtss_phy_ts_ptp_delaym_type_t	988
8.32.4.8 vtss_phy_ts_y1731_oam_delaym_type_t	988
8.32.4.9 vtss_phy_ts_ietf_mpls_ach_oam_delaym_type_t	989
8.32.4.10 vtss_phy_ts_ietf_mpls_ach_oam_ts_format_t	989
8.32.4.11 vtss_phy_ts_action_format	990
8.32.4.12 vtss_phy_ts_clockfreq_t	990
8.32.4.13 vtss_phy_ts_clock_src_t	990
8.32.4.14 vtss_phy_ts_rxtimestamp_pos_t	991
8.32.4.15 vtss_phy_ts_rxtimestamp_len_t	991
8.32.4.16 vtss_phy_ts_fifo_mode_t	992
8.32.4.17 vtss_phy_ts_fifo_timestamp_len_t	992
8.32.4.18 vtss_phy_ts_tc_op_mode_t	992

8.32.4.19 <code>vtss_phy_ts_nphase_sampler_t</code>	993
8.32.4.20 <code>vtss_phy_ts_ptp_message_type_t</code>	993
8.32.5 Function Documentation	994
8.32.5.1 <code>vtss_phy_ts_alt_clock_saved_get()</code>	994
8.32.5.2 <code>vtss_phy_ts_alt_clock_mode_get()</code>	994
8.32.5.3 <code>vtss_phy_ts_alt_clock_mode_set()</code>	994
8.32.5.4 <code>vtss_phy_ts_pps_conf_set()</code>	995
8.32.5.5 <code>vtss_phy_ts_pps_conf_get()</code>	995
8.32.5.6 <code>vtss_phy_ts_ingress_latency_set()</code>	996
8.32.5.7 <code>vtss_phy_ts_ingress_latency_get()</code>	996
8.32.5.8 <code>vtss_phy_ts_egress_latency_set()</code>	997
8.32.5.9 <code>vtss_phy_ts_egress_latency_get()</code>	997
8.32.5.10 <code>vtss_phy_ts_path_delay_set()</code>	998
8.32.5.11 <code>vtss_phy_ts_path_delay_get()</code>	998
8.32.5.12 <code>vtss_phy_ts_delay_asymmetry_set()</code>	999
8.32.5.13 <code>vtss_phy_ts_delay_asymmetry_get()</code>	999
8.32.5.14 <code>vtss_phy_ts_ptptime_set()</code>	1000
8.32.5.15 <code>vtss_phy_ts_ptptime_set_done()</code>	1000
8.32.5.16 <code>vtss_phy_ts_ptptime_arm()</code>	1001
8.32.5.17 <code>vtss_phy_ts_ptptime_get()</code>	1001
8.32.5.18 <code>vtss_phy_ts_load_ptptime_get()</code>	1002
8.32.5.19 <code>vtss_phy_ts_sertod_set()</code>	1002
8.32.5.20 <code>vtss_phy_ts_sertod_get()</code>	1002
8.32.5.21 <code>vtss_phy_ts_loadpulse_delay_set()</code>	1003
8.32.5.22 <code>vtss_phy_ts_loadpulse_delay_get()</code>	1003
8.32.5.23 <code>vtss_phy_ts_clock_rateadj_set()</code>	1004
8.32.5.24 <code>vtss_phy_ts_clock_rateadj_get()</code>	1004
8.32.5.25 <code>vtss_phy_ts_clock_rateadj_ppm_set()</code>	1005
8.32.5.26 <code>vtss_phy_ts_clock_rateadj_ppm_get()</code>	1005
8.32.5.27 <code>vtss_phy_ts_ptptime_adj1ns()</code>	1005

8.32.5.28 vtss_phy_ts_timeofday_offset_set()	1006
8.32.5.29 vtss_phy_ts_ongoing_adjustment()	1006
8.32.5.30 vtss_phy_ts_ltc_freq_synth_pulse_set()	1007
8.32.5.31 vtss_phy_ts_ltc_freq_synth_pulse_get()	1007
8.32.5.32 vtss_phy_daisy_conf_set()	1008
8.32.5.33 vtss_phy_daisy_conf_get()	1008
8.32.5.34 vtss_phy_ts_fifo_sig_set()	1008
8.32.5.35 vtss_phy_ts_fifo_sig_get()	1009
8.32.5.36 vtss_phy_ts_fifo_empty()	1009
8.32.5.37 vtss_phy_ts_fifo_read_install()	1010
8.32.5.38 vtss_phy_ts_fifo_read_cb_get()	1010
8.32.5.39 vtss_phy_ts_ingress_engine_init()	1011
8.32.5.40 vtss_phy_ts_ingress_engine_init_conf_get()	1012
8.32.5.41 vtss_phy_ts_ingress_engine_clear()	1012
8.32.5.42 vtss_phy_ts_egress_engine_init()	1013
8.32.5.43 vtss_phy_ts_egress_engine_init_conf_get()	1013
8.32.5.44 vtss_phy_ts_egress_engine_clear()	1014
8.32.5.45 vtss_phy_ts_ingress_engine_conf_set()	1014
8.32.5.46 vtss_phy_ts_ingress_engine_conf_get()	1015
8.32.5.47 vtss_phy_ts_egress_engine_conf_set()	1015
8.32.5.48 vtss_phy_ts_egress_engine_conf_get()	1016
8.32.5.49 vtss_phy_ts_ingress_engine_action_set()	1016
8.32.5.50 vtss_phy_ts_ingress_engine_action_get()	1017
8.32.5.51 vtss_phy_ts_egress_engine_action_set()	1018
8.32.5.52 vtss_phy_ts_egress_engine_action_get()	1018
8.32.5.53 vtss_phy_ts_event_enable_set()	1019
8.32.5.54 vtss_phy_ts_event_enable_get()	1019
8.32.5.55 vtss_phy_ts_event_poll()	1019
8.32.5.56 vtss_phy_ts_stats_get()	1020
8.32.5.57 vtss_phy_ts_correction_overflow_get()	1020

8.32.5.58 vtss_phy_ts_mode_set()	1021
8.32.5.59 vtss_phy_ts_mode_get()	1021
8.32.5.60 vtss_phy_ts_init()	1022
8.32.5.61 vtss_phy_ts_init_conf_get()	1022
8.32.5.62 vtss_phy_ts_nphase_status_get()	1023
8.32.5.63 vtss_phy_ts_hiacc_set()	1023
8.32.5.64 vtss_phy_ts_hiacc_get()	1024
8.32.5.65 vtss_phy_ts_block_soft_reset()	1024
8.32.5.66 vtss_phy_ts_new_spi_mode_set()	1025
8.32.5.67 vtss_phy_ts_new_spi_mode_get()	1025
8.32.5.68 vtss_phy_ts_phy_oper_mode_change()	1026
8.32.5.69 vtss_phy_1588_csr_reg_write()	1026
8.32.5.70 vtss_phy_1588_csr_reg_read()	1027
8.32.5.71 vtss_phy_ts_status_check()	1027
8.32.5.72 vtss_phy_ts_10g_extended_fifo_sync()	1028
8.32.5.73 vtss_phy_ts_10g_fifo_sync()	1028
8.32.5.74 vtss_phy_ts_bypass_clear()	1028
8.32.5.75 vtss_phy_ts_viper_fifo_reset()	1029
8.32.5.76 vtss_phy_ts_tesla_tsp_fifo_sync()	1029
8.32.5.77 vtss_phy_1g_ts_fifo_sync()	1030
8.32.5.78 vtss_phy_1588_debug_reg_read()	1030
8.32.5.79 vtss_phy_ts_flow_clear_cf_set()	1031
8.33 vtss_api/include/vtss_port_api.h File Reference	1031
8.33.1 Detailed Description	1034
8.33.2 Macro Definition Documentation	1034
8.33.2.1 CHIP_PORT_UNUSED	1034
8.33.2.2 VTSS_FRAME_GAP_DEFAULT	1034
8.33.2.3 VTSS_MAX_FRAME_LENGTH_STANDARD	1035
8.33.2.4 VTSS_MAX_FRAME_LENGTH_MAX [1/2]	1035
8.33.2.5 VTSS_MAX_FRAME_LENGTH_MAX [2/2]	1035

8.33.3 Enumeration Type Documentation	1035
8.33.3.1 <code>vtss_miim_controller_t</code>	1035
8.33.3.2 <code>vtss_internal_bw_t</code>	1036
8.33.3.3 <code>vtss_port_clause_37_remote_fault_t</code>	1036
8.33.3.4 <code>vtss_port_max_tags_t</code>	1036
8.33.3.5 <code>vtss_port_loop_t</code>	1037
8.33.3.6 <code>vtss_port_forward_t</code>	1037
8.33.4 Function Documentation	1037
8.33.4.1 <code>vtss_port_map_set()</code>	1037
8.33.4.2 <code>vtss_port_map_get()</code>	1038
8.33.4.3 <code>vtss_port_clause_37_control_get()</code>	1038
8.33.4.4 <code>vtss_port_clause_37_control_set()</code>	1039
8.33.4.5 <code>vtss_port_conf_set()</code>	1039
8.33.4.6 <code>vtss_port_conf_get()</code>	1040
8.33.4.7 <code>vtss_port_status_get()</code>	1040
8.33.4.8 <code>vtss_port_counters_update()</code>	1040
8.33.4.9 <code>vtss_port_counters_clear()</code>	1041
8.33.4.10 <code>vtss_port_counters_get()</code>	1041
8.33.4.11 <code>vtss_port_basic_counters_get()</code>	1042
8.33.4.12 <code>vtss_port_forward_state_get()</code>	1042
8.33.4.13 <code>vtss_port_forward_state_set()</code>	1043
8.33.4.14 <code>vtss_port_ifh_conf_set()</code>	1043
8.33.4.15 <code>vtss_port_ifh_conf_get()</code>	1043
8.33.4.16 <code>vtss_miim_read()</code>	1044
8.33.4.17 <code>vtss_miim_write()</code>	1044
8.33.4.18 <code>vtss_port_mmd_read()</code>	1045
8.33.4.19 <code>vtss_port_mmd_read_inc()</code>	1045
8.33.4.20 <code>vtss_port_mmd_write()</code>	1046
8.33.4.21 <code>vtss_port_mmd_masked_write()</code>	1046
8.33.4.22 <code>vtss_mmd_read()</code>	1047

8.33.4.23 <code>vtss_mmd_write()</code>	1047
8.34 <code>vtss_api/include/vtss_qos_api.h</code> File Reference	1048
8.34.1 Detailed Description	1050
8.34.2 Macro Definition Documentation	1050
8.34.2.1 <code>VTSS_PORT_POLICERS</code>	1050
8.34.2.2 <code>VTSS_PORT_POLICER_CPU_QUEUES</code>	1050
8.34.2.3 <code>VTSS_QCL_IDS</code>	1051
8.34.2.4 <code>VTSS_QCL_ID_START</code>	1051
8.34.2.5 <code>VTSS_QCL_ID_END</code>	1051
8.34.2.6 <code>VTSS_QCL_ARRAY_SIZE</code>	1051
8.34.2.7 <code>VTSS_QCE_ID_LAST</code>	1051
8.34.3 Enumeration Type Documentation	1051
8.34.3.1 <code>vtss_tag_remark_mode_t</code>	1051
8.34.3.2 <code>vtss_dscp_mode_t</code>	1052
8.34.3.3 <code>vtss_dscp_emode_t</code>	1052
8.34.3.4 <code>vtss_qce_type_t</code>	1052
8.34.4 Function Documentation	1053
8.34.4.1 <code>vtss_qos_conf_get()</code>	1053
8.34.4.2 <code>vtss_qos_conf_set()</code>	1053
8.34.4.3 <code>vtss_qos_port_conf_get()</code>	1054
8.34.4.4 <code>vtss_qos_port_conf_set()</code>	1054
8.34.4.5 <code>vtss_qce_init()</code>	1055
8.34.4.6 <code>vtss_qce_add()</code>	1055
8.34.4.7 <code>vtss_qce_del()</code>	1055
8.35 <code>vtss_api/include/vtss_rab_api.h</code> File Reference	1057
8.35.1 Detailed Description	1057
8.36 <code>vtss_api/include/vtss_security_api.h</code> File Reference	1057
8.36.1 Detailed Description	1059
8.36.2 Macro Definition Documentation	1059
8.36.2.1 <code>VTSS_ACE_ID_LAST</code>	1060

8.36.2.2 VTSS_PORT_NO_ANY	1060
8.36.3 Enumeration Type Documentation	1060
8.36.3.1 vtss_auth_state_t	1060
8.36.3.2 vtss_ace_type_t	1060
8.36.3.3 vtss_ace_bit_t	1061
8.36.4 Function Documentation	1061
8.36.4.1 vtss_auth_port_state_get()	1061
8.36.4.2 vtss_auth_port_state_set()	1062
8.36.4.3 vtss_acl_policer_conf_get()	1062
8.36.4.4 vtss_acl_policer_conf_set()	1063
8.36.4.5 vtss_acl_port_conf_get()	1063
8.36.4.6 vtss_acl_port_conf_set()	1063
8.36.4.7 vtss_acl_port_counter_get()	1064
8.36.4.8 vtss_acl_port_counter_clear()	1064
8.36.4.9 vtss_ace_init()	1065
8.36.4.10 vtss_ace_add()	1065
8.36.4.11 vtss_ace_del()	1066
8.36.4.12 vtss_ace_counter_get()	1066
8.36.4.13 vtss_ace_counter_clear()	1066
8.37 vtss_api/include/vtss_sfi4_api.h File Reference	1067
8.37.1 Detailed Description	1067
8.38 vtss_api/include/vtss_sync_api.h File Reference	1067
8.38.1 Detailed Description	1068
8.38.2 Macro Definition Documentation	1068
8.38.2.1 VTSS_SYNCE_CLK_A	1068
8.38.2.2 VTSS_SYNCE_CLK_B	1068
8.38.2.3 VTSS_SYNCE_CLK_MAX	1069
8.38.3 Enumeration Type Documentation	1069
8.38.3.1 vtss_sync_divider_t	1069
8.38.4 Function Documentation	1069

8.38.4.1 vtss_sync_clock_out_set()	1069
8.38.4.2 vtss_sync_clock_out_get()	1070
8.38.4.3 vtss_sync_clock_in_set()	1070
8.38.4.4 vtss_sync_clock_in_get()	1071
8.39 vtss_api/include/vtss_tfi5_api.h File Reference	1071
8.39.1 Detailed Description	1071
8.40 vtss_api/include/vtss_ts_api.h File Reference	1071
8.40.1 Detailed Description	1074
8.40.2 Function Documentation	1074
8.40.2.1 vtss_ts_timeofday_set()	1074
8.40.2.2 vtss_ts_timeofday_set_delta()	1075
8.40.2.3 vtss_ts_timeofday_offset_set()	1075
8.40.2.4 vtss_ts_adjtimer_one_sec()	1076
8.40.2.5 vtss_ts_ongoing_adjustment()	1076
8.40.2.6 vtss_ts_timeofday_get()	1076
8.40.2.7 vtss_ts_timeofday_next_pps_get()	1077
8.40.2.8 vtss_ts_adjtimer_set()	1077
8.40.2.9 vtss_ts_adjtimer_get()	1078
8.40.2.10 vtss_ts_freq_offset_get()	1078
8.40.2.11 vtss_ts_external_clock_mode_get()	1078
8.40.2.12 vtss_ts_external_clock_mode_set()	1079
8.40.2.13 vtss_ts_external_clock_saved_get()	1079
8.40.2.14 vtss_ts_ingress_latency_set()	1080
8.40.2.15 vtss_ts_ingress_latency_get()	1080
8.40.2.16 vtss_ts_p2p_delay_set()	1081
8.40.2.17 vtss_ts_p2p_delay_get()	1081
8.40.2.18 vtss_ts_egress_latency_set()	1081
8.40.2.19 vtss_ts_egress_latency_get()	1082
8.40.2.20 vtss_ts_delay_asymmetry_set()	1082
8.40.2.21 vtss_ts_delay_asymmetry_get()	1083

8.40.2.22 vtss_ts_operation_mode_set()	1083
8.40.2.23 vtss_ts_operation_mode_get()	1084
8.40.2.24 vtss_ts_internal_mode_set()	1084
8.40.2.25 vtss_ts_internal_mode_get()	1084
8.40.2.26 vtss_tx_timestamp_update()	1085
8.40.2.27 vtss_rx_timestamp_get()	1085
8.40.2.28 vtss_rx_timestamp_id_release()	1086
8.40.2.29 vtss_rx_master_timestamp_get()	1086
8.40.2.30 vtss_tx_timestamp_idx_alloc()	1086
8.40.2.31 vtss_timestamp_age()	1087
8.40.2.32 vtss_ts_status_change()	1087
8.41 vtss_api/include/vtss_upi_api.h File Reference	1088
8.41.1 Detailed Description	1088
8.42 vtss_api/include/vtss_wis_api.h File Reference	1088
8.42.1 Detailed Description	1093
8.42.2 Macro Definition Documentation	1093
8.42.2.1 VTSS_EWIS_SEF_EV	1093
8.42.2.2 VTSS_EWIS_FPLM_EV	1094
8.42.2.3 VTSS_EWIS_FAIS_EV	1094
8.42.2.4 VTSS_EWIS_LOF_EV	1094
8.42.2.5 VTSS_EWIS_LOS_EV	1094
8.42.2.6 VTSS_EWIS_RDIL_EV	1094
8.42.2.7 VTSS_EWIS_AISL_EV	1095
8.42.2.8 VTSS_EWIS_LCDP_EV	1095
8.42.2.9 VTSS_EWIS_PLMP_EV	1095
8.42.2.10 VTSS_EWIS_AISP_EV	1095
8.42.2.11 VTSS_EWIS_LOPP_EV	1095
8.42.2.12 VTSS_EWIS_MODULE_EV	1096
8.42.2.13 VTSS_EWIS_TXLOL_EV	1096
8.42.2.14 VTSS_EWIS_RXLOL_EV	1096

8.42.2.15 VTSS_EWIS_LOPC_EV	1096
8.42.2.16 VTSS_EWIS_UNEQP_EV	1096
8.42.2.17 VTSS_EWIS_FEUNEQP_EV	1097
8.42.2.18 VTSS_EWIS_FERDIP_EV	1097
8.42.2.19 VTSS_EWIS_REIL_EV	1097
8.42.2.20 VTSS_EWIS_REIP_EV	1097
8.42.2.21 VTSS_EWIS_HIGH_BER_EV	1097
8.42.2.22 VTSS_EWIS_PCS_RECEIVE_FAULT_PEND	1098
8.42.2.23 VTSS_EWIS_B1_NZ_EV	1098
8.42.2.24 VTSS_EWIS_B2_NZ_EV	1098
8.42.2.25 VTSS_EWIS_B3_NZ_EV	1098
8.42.2.26 VTSS_EWIS_REIL_NZ_EV	1098
8.42.2.27 VTSS_EWIS_REIP_NZ_EV	1099
8.42.2.28 VTSS_EWIS_B1_THRESH_EV	1099
8.42.2.29 VTSS_EWIS_B2_THRESH_EV	1099
8.42.2.30 VTSS_EWIS_B3_THRESH_EV	1099
8.42.2.31 VTSS_EWIS_REIL_THRESH_EV	1099
8.42.2.32 VTSS_EWIS_REIP_THRESH_EV	1100
8.42.3 Typedef Documentation	1100
8.42.3.1 vtss_ewis_static_conf_t	1100
8.42.3.2 vtss_ewis_event_t	1100
8.42.4 Enumeration Type Documentation	1100
8.42.4.1 vtss_ewis_tti_mode_t	1100
8.42.4.2 vtss_ewis_perf_cntr_mode_t	1101
8.42.4.3 vtss_ewis_mode_t	1101
8.42.4.4 vtss_ewis_test_pattern_s	1101
8.42.4.5 vtss_ewis_prbs31_err_inj_t	1102
8.42.5 Function Documentation	1102
8.42.5.1 vtss_ewis_event_enable()	1102
8.42.5.2 vtss_ewis_event_poll()	1103

8.42.5.3 vtss_ewis_event_poll_without_mask()	1103
8.42.5.4 vtss_ewis_event_force()	1104
8.42.5.5 vtss_ewis_static_conf_get()	1104
8.42.5.6 vtss_ewis_force_conf_set()	1105
8.42.5.7 vtss_ewis_force_conf_get()	1105
8.42.5.8 vtss_ewis_tx_oh_set()	1105
8.42.5.9 vtss_ewis_tx_oh_get()	1106
8.42.5.10 vtss_ewis_tx_oh_passthru_set()	1106
8.42.5.11 vtss_ewis_tx_oh_passthru_get()	1107
8.42.5.12 vtss_ewis_mode_set()	1107
8.42.5.13 vtss_ewis_mode_get()	1108
8.42.5.14 vtss_ewis_reset()	1108
8.42.5.15 vtss_ewis_cons_act_set()	1109
8.42.5.16 vtss_ewis_cons_act_get()	1109
8.42.5.17 vtss_ewis_section_txi_set()	1109
8.42.5.18 vtss_ewis_section_txi_get()	1110
8.42.5.19 vtss_ewis_exp_sl_set()	1110
8.42.5.20 vtss_ewis_path_txi_set()	1111
8.42.5.21 vtss_ewis_path_txi_get()	1111
8.42.5.22 vtss_ewis_test_mode_set()	1112
8.42.5.23 vtss_ewis_test_mode_get()	1112
8.42.5.24 vtss_ewis_prbs31_err_inj_set()	1113
8.42.5.25 vtss_ewis_test_counter_get()	1113
8.42.5.26 vtss_ewis_defects_get()	1114
8.42.5.27 vtss_ewis_status_get()	1114
8.42.5.28 vtss_ewis_section_acti_get()	1114
8.42.5.29 vtss_ewis_path_acti_get()	1115
8.42.5.30 vtss_ewis_counter_get()	1115
8.42.5.31 vtss_ewis_perf_get()	1116
8.42.5.32 vtss_ewis_counter_threshold_set()	1116
8.42.5.33 vtss_ewis_counter_threshold_get()	1117
8.42.5.34 vtss_ewis_perf_mode_set()	1117
8.42.5.35 vtss_ewis_perf_mode_get()	1117
8.43 vtss_api/include/vtss_xaui_api.h File Reference	1118
8.43.1 Detailed Description	1118
8.44 vtss_api/include/vtss_xfi_api.h File Reference	1118
8.44.1 Detailed Description	1118
Index	1119

Chapter 1

Ethernet Virtual Connections

Ethernet Virtual Connections (EVCs) are based on Provider Bridging. It is possible to setup MEF compliant EVCs including Ingress Bandwidth Profiles. The normal procedure for configuring EVCs is:

- 1) Setup VLAN port types using [vtss_vlan_port_conf_set\(\)](#).
- 2) Setup VLAN membership using [vtss_vlan_port_members_set\(\)](#).
- 3) Add EVCs using [vtss_evc_add\(\)](#), specifying the NNI ports and VID of the outer tag.
- 4) Add ECEs using [vtss_ece_add\(\)](#), specifying the UNI ports and frames mapping to the EVCs.
- 5) Setup EVC policers using [vtss_evc_policer_conf_set\(\)](#).

1.1 Port Configuration

The following EVC functions are available per port.

- [vtss_evc_port_conf_get\(\)](#) is used to get the EVC port configuration.
- [vtss_evc_port_conf_set\(\)](#) is used to set the EVC port configuration.

1.2 Policer Configuration

EVC policers are Ingress Bandwidth Profiles applied to frames classified to an EVC. Each EVC policer is identified by a policer ID ([vtss_evc_policer_id_t](#)). The following EVC policer functions are available:

- [vtss_evc_policer_conf_get\(\)](#) is used to get the EVC policer configuration.
- [vtss_evc_policer_conf_set\(\)](#) is used to set the EVC policer configuration.

The following policer IDs are reserved for special purposes and can not be changed:

- [VTSS_EVC_POLICER_ID_DISCARD](#) used for an EVC/ECE indicates that all frames are discarded.
- [VTSS_EVC_POLICER_ID_NONE](#) used for an EVC/ECE indicates that all frames are forwarded.
- [VTSS_EVC_POLICER_ID_EVC](#) used for an ECE indicates that the policer of the EVC is used.

By default, all EVC policers are disabled.

1.3 EVCs

Each EVC rule is identified by an EVC ID ([vtss_evc_id_t](#)). The following functions are available:

- [vtss_evc_add\(\)](#) is used to add an EVC rule.
- [vtss_evc_del\(\)](#) is used to delete an EVC rule.
- [vtss_evc_get\(\)](#) is used to get an EVC rule.

The [vtss_evc_conf_t](#) structure used when adding an EVC rule includes the following:

- NNI port list.
- VLAN ID used in outer tag on NNI ports.
- Internal/classified VLAN ID used for forwarding and learning.

By default, no EVCs are setup.

1.4 ECEs

Each EVC Control Entry (ECE) is identified by an ECE ID used for identification and ordering of ECEs. The following functions are available:

- [vtss_ece_add\(\)](#) is used to add an ECE.
- [vtss_ece_del\(\)](#) is used to delete an ECE.

The [vtss_ece_t](#) structure used when adding an ECE includes the following fields:

- ECE ID ([vtss_ece_id_t](#)) used for identifying the rule.
- ECE key fields ([vtss_ece_key_t](#)), including:
 - UNI port list.
 - Frame type and frame specific fields.
- ECE action fields ([vtss_ece_action_t](#)), including:
 - EVC ID ([vtss_evc_id_t](#)) to which the ECE is mapping.
 - Direction ([vtss_ece_dir_t](#)) determining if UNI-to-NNI, NNI-to-UNI or bidirectional processing is done.
 - Tag pop count ([vtss_ece_pop_tag_t](#)).
 - Outer tag properties ([vtss_ece_outer_tag_t](#)) determining the PCP and DEI values.
 - ACL policy number ([vtss_acl_policy_no_t](#)) for ACL rule processing.

By default, no ECEs are setup.

1.5 EVC Statistics

EVC statistics are available per EVC ID and UNI>NNI port.

- [`vtss_evc_counters_get\(\)`](#) is used to get the counters for an EVC and port.
- [`vtss_evc_counters_clear\(\)`](#) is used to clear the counters for an EVC and port.

1.6 ECE Statistics

ECE statistics are available per ECE ID and UNI>NNI port.

- [`vtss_ece_counters_get\(\)`](#) is used to get the counters for an EVC and port.
- [`vtss_ece_counters_clear\(\)`](#) is used to clear the counters for an EVC and port.

Chapter 2

Layer 2

The Layer 2 functions are used to control basic switching features.

2.1 MAC Address Table

The MAC address table functions are used to control the Layer 2 forwarding database. Each entry is identified by VLAN ID and MAC address ([vtss_vid_mac_t](#)). The following MAC address table functions are available:

- [vtss_mac_table_add\(\)](#) is used to add a static entry.
- [vtss_mac_table_del\(\)](#) is used to delete a static entry.
- [vtss_mac_table_get\(\)](#) is used to lookup a specific entry.
- [vtss_mac_table_get_next\(\)](#) is used to get the next entry for table traversal.
- [vtss_mac_table_age_time_get\(\)](#) is used to get the age time.
- [vtss_mac_table_age_time_set\(\)](#) is used to set the age time.
- [vtss_mac_table_age\(\)](#) is used for manual age scan.
- [vtss_mac_table_vlan_age\(\)](#) is used for manual age scan per VLAN.
- [vtss_mac_table_flush\(\)](#) is used to flush all dynamic entries.
- [vtss_mac_table_port_flush\(\)](#) is used to flush dynamic entries per port.
- [vtss_mac_table_vlan_flush\(\)](#) is used to flush dynamic entries per VLAN.
- [vtss_mac_table_vlan_port_flush\(\)](#) is used to flush dynamic entries per VLAN and port.
- [vtss_mac_table_upsid_flush\(\)](#) is used to flush dynamic entries per UPSID.
- [vtss_mac_table_upsid_upspn_flush\(\)](#) is used to flush dynamic entries per (UPSID, UPSPN).
- [vtss_mac_table_status_get\(\)](#) is used to poll for MAC address table change events.
- [vtss_learn_port_mode_get\(\)](#) is used to get the learn mode per port.
- [vtss_learn_port_mode_set\(\)](#) is used to set the learn mode per port.

By default, automatic learning and ageing is enabled.

2.2 Operational State

When the application detects link state changes, the operational port state must be setup. This ensures that frames are forwarded to operational ports only. The following functions are available:

- [vtss_port_state_get\(\)](#) is used to get the forwarding state for each port.
- [vtss_port_state_set\(\)](#) is used to set the forwarding state for each port.

By default, all ports are down.

2.3 Spanning Tree

The following Spanning Tree functions are available:

- [vtss_stp_port_state_get\(\)](#) is used to get the STP state for a port.
- [vtss_stp_port_state_set\(\)](#) is used to set the STP state for a port.
- [vtss_mstp_vlan_msti_get\(\)](#) is used to get the mapping from VLAN to MSTP instance.
- [vtss_mstp_vlan_msti_set\(\)](#) is used to set the mapping from VLAN to MSTP instance.
- [vtss_mstp_port_msti_state_get\(\)](#) is used to get the MSTP state per MSTP instance and port.
- [vtss_mstp_port_msti_state_set\(\)](#) is used to set the MSTP state per MSTP instance and port.

By default, all ports are in STP forwarding mode.

By default, all VLANs map to the first MSTP instance and all ports are forwarding for that instance.

2.4 VLAN

Basic VLAN classification and tag preservation works as follows:

- Untagged and priority-tagged frames received on a port are classified to the Port VLAN ID (PVID).
- VLAN-tagged frames received on a VLAN unaware port are classified to the PVID and the tag is preserved.
- VLAN-tagged frames received on a VLAN aware port are classified to the VLAN ID in the tag and the tag is stripped.

In the egress direction, a tag with the classified VID will be added if the Untagged VID (UVID) of the port is not [VTSS_VID_ALL](#) and the classified VID is different from the UVID of the port. Setting the UVID to [VTSS_VID_NULL](#) will cause all frames to have a tag added.

The following VLAN functions are available:

- [vtss_vlan_conf_get\(\)](#) is used to get the global VLAN configuration.

- `vtss_vlan_conf_set()` is used to set the global VLAN configuration.
- `vtss_vlan_port_conf_get()` is used to get the VLAN port configuration.
- `vtss_vlan_port_conf_set()` is used to set the VLAN port configuration.
- `vtss_vlan_tx_tag_get()` is used to get the advanced tagging configuration for a VLAN.
- `vtss_vlan_tx_tag_set()` is used to set the advanced tagging configuration for a VLAN.
- `vtss_vlan_port_members_get()` is used to get the VLAN port members.
- `vtss_vlan_port_members_set()` is used to set the VLAN port members.
- `vtss_vlan_vid_conf_get()` is used to get VLAN configuration.
- `vtss_vlan_vid_conf_set()` is used to set VLAN configuration.

By default, all ports are VLAN unaware with port VLAN ID 1 and members of VLAN 1 only.

2.5 VLAN Classification List

Advanced VLAN classification rules can be done using the VLAN Classification List (VCL). Each VLAN Classification Entry (VCE) is identified by a VCE ID (`vtss_vce_id_t`). The following VCL functions are available:

- `vtss_vcl_port_conf_get()` is used to get the VCL port configuration.
- `vtss_vcl_port_conf_set()` is used to set the VCL port configuration.
- `vtss_vce_init()` is used to initialize a VCE to default values.
- `vtss_vce_add()` is used to add or modify a VCE.
- `vtss_vce_del()` is used to delete a VCE.

The VCEs are ordered in a list of rules based on the VCE IDs. When adding a rule, the VCE ID of the rule and the VCE ID of the next rule in the list must be specified. A special value `VTSS_VCE_ID_LAST` is used to specify that the rule must be added at the end of the list.

Each VCE includes a key structure (`vtss_vce_key_t`) with fields used for matching received frames and an action structure (`vtss_vce_action_t`) with the classified VLAN ID.

By default, no VCE rules are setup.

2.6 VLAN Translation

VLAN translation can be used on ports connecting two VLAN domains. If multiple ports are used for the connection (e.g. link aggregation), the ports can be grouped. VLAN translation rules can be added to each group. The following functions are available:

- `vtss_vlan_trans_group_to_port_get()` is used to get the ports of a translation group.
- `vtss_vlan_trans_group_to_port_set()` is used to set the ports of a translation group.
- `vtss_vlan_trans_group_add()` is used to add a VLAN translation to a group.
- `vtss_vlan_trans_group_del()` is used to delete a VLAN translation from a group.

By default, no VLAN translation rules are setup.

2.7 Port Isolation

Port isolation can be used to restrict forwarding between ports. If isolation is enabled for both the ingress port and the classified VLAN of a frame, the frame can not be forwarded to other isolated ports. The following functions are available:

- [vtss_isolated_vlan_get\(\)](#) is used to get the isolation mode for a VLAN.
- [vtss_isolated_vlan_set\(\)](#) is used to set the isolation mode for a VLAN.
- [vtss_isolated_port_members_get\(\)](#) is used to get the isolated port members.
- [vtss_isolated_port_members_set\(\)](#) is used to get the isolated port members.

By default, port isolation is disabled for all ports and VLANs.

2.8 Private VLAN

Private VLANs can be used to divide ports into groups and restrict forwarding independently of traditional VLANs. Each PVLAN is identified by a PVLAN number ([vtss_pvlan_no_t](#)). Forwarding between two ports is only possible if both ports are included in at least one PVLAN. The following functions are available:

- [vtss_pvlan_port_members_get\(\)](#) is used to get the port members of PVLAN.
- [vtss_pvlan_port_members_set\(\)](#) is used to set the port members of PVLAN.

By default, all ports are included in PVLAN 1.

2.9 Asymmetric Private VLAN

Asymmetric Private VLANs can be used to restrict forwarding independently of traditional VLANs. For each ingress port it is possible to define which other egress ports it is allowed to forward to. The following functions are available:

- [vtss_apvlan_port_members_get\(\)](#) is used to get the egress port members for an ingress port.
- [vtss_apvlan_port_members_set\(\)](#) is used to set the egress port members for an ingress port.

By default, all ports are allowed to forward to all other ports.

2.10 Destination Port Groups

Destination Port Groups can be used to ensure that frames are forwarded to all ports in the same group.

- [vtss_dgroup_port_conf_get\(\)](#) is used to get the destination group for a port.
- [vtss_dgroup_port_conf_set\(\)](#) is used to set the destination group for a port.

By default, each port is in a destination group identical to the port number.

2.11 sFlow

The sFlow functions can be used to sample frame flows.

- [vtss_sflow_port_conf_get\(\)](#) is used to get the sFlow port configuration.
- [vtss_sflow_port_conf_set\(\)](#) is used to set the sFlow port configuration.
- [vtss_sflow_sampling_rate_convert\(\)](#) converts desired sampling rate to actual sampling rate.

By default, sFlow is disabled on all ports.

2.12 Link Aggregation

A link aggregation forms one logical link based on multiple physical ports. Each link aggregation is identified by an aggregation number ([vtss_aggr_no_t](#)). The aggregation mode of the switch determines how traffic forwarded to link aggregations is distributed on the physical ports. The following functions are available:

- [vtss_aggr_port_members_get\(\)](#) is used to get the aggregation port members.
- [vtss_aggr_port_members_set\(\)](#) is used to set the aggregation port members.
- [vtss_aggr_mode_get\(\)](#) is used to get the aggregation mode.
- [vtss_aggr_mode_set\(\)](#) is used to set the aggregation mode.

By default, no link aggregations exist.

2.13 Global Link Aggregation

A global link aggregation forms one logical link based on ports in a stack. Each global link aggregation is identified by an GLAG number ([vtss_glag_no_t](#)).

- [vtss_aggr_glag_members_get\(\)](#) is used to get the local GLAG port members.
- [vtss_vstax_glag_get\(\)](#) is used to get the GLAG port members.
- [vtss_vstax_glag_set\(\)](#) is used to set the GLAG port members.

By default, no global link aggregations exist.

2.14 Mirroring

Mirroring can be used to copy frames to a monitor port for network troubleshooting purposes. The following functions are available:

- [vtss_mirror_conf_get\(\)](#) is used to get the mirror configuration.
- [vtss_mirror_conf_set\(\)](#) is used to set the mirror configuration.
- [vtss_mirror_monitor_port_get\(\)](#) is used to get the mirror monitor port.
- [vtss_mirror_monitor_port_set\(\)](#) is used to set the mirror monitor port.
- [vtss_mirror_ingress_ports_get\(\)](#) is used to get the ingress mirroring port members.
- [vtss_mirror_ingress_ports_set\(\)](#) is used to set the ingress mirroring port members.
- [vtss_mirror_egress_ports_get\(\)](#) is used to get the egress mirroring port members.
- [vtss_mirror_egress_ports_set\(\)](#) is used to set the egress mirroring port members.
- [vtss_mirror_cpu_ingress_get\(\)](#) is used to get the CPU ingress mirror mode.
- [vtss_mirror_cpu_ingress_set\(\)](#) is used to set the CPU ingress mirror mode.
- [vtss_mirror_cpu_egress_get\(\)](#) is used to get the CPU egress mirror mode.
- [vtss_mirror_cpu_egress_set\(\)](#) is used to set the CPU egress mirror mode.

By default, mirroring is disabled for all ports.

2.15 Flooding Control

Forwarding of frames with unknown destination MAC addresses can be controlled per egress port:

- [vtss_uc_flood_members_get\(\)](#) is used to get the unicast flooding port members.
- [vtss_uc_flood_members_set\(\)](#) is used to set the unicast flooding port members.
- [vtss_mc_flood_members_get\(\)](#) is used to get the non-IP multicast flooding port members.
- [vtss_mc_flood_members_set\(\)](#) is used to set the non-IP multicast flooding port members.

By default, unicast and non-IP multicast flooding is enabled for all ports.

2.16 IPv4 Multicast

Forwarding of IPv4 multicast frames may be restricted based on IGMP snooping:

- [vtss_ipv4_mc_flood_members_get\(\)](#) is used to get IPv4 multicast flooding port members.
- [vtss_ipv4_mc_flood_members_set\(\)](#) is used to set IPv4 multicast flooding port members.

By default, IPv4 multicast flooding is enabled for all ports.

2.17 IPv6 Multicast

Forwarding of IPv6 multicast frames may be restricted based on MLD snooping:

- `vtss_ipv6_mc_flood_members_get()` is used to get the IPv6 multicast flooding port members.
- `vtss_ipv6_mc_flood_members_set()` is used to set the IPv6 multicast flooding port members.
- `vtss_ipv6_mc_ctrl_flood_get()` is used to get the IPv6 multicast control flooding mode.
- `vtss_ipv6_mc_ctrl_flood_set()` is used to set the IPv6 multicast control flooding mode.

By default, IPv6 multicast flooding is enabled for all ports.

2.18 Ethernet Protection Switching

Linear Ethernet Protection Switching can be controlled using EPS functions. Ports may be setup with 1:1 or 1+1 protection.

- `vtss_eps_port_conf_get()` is used to get the EPS port configuration.
- `vtss_eps_port_conf_set()` is used to set the EPS port configuration.
- `vtss_eps_port_selector_get()` is used to get the protection selector state.
- `vtss_eps_port_selector_set()` is used to set the protection selector state.

By default, all ports are unprotected.

2.19 Ethernet Ring Protection Switching

Ethernet Ring Protection Switching can be controlled using ERPS functions. Each ring is identified by an ERPS instance number. Each VLAN can be enabled for one or multiple ERPS instances. The forwarding mode can be controlled per ERPS instance and port.

- `vtss_erps_vlan_member_get()` is used to get the ERPS member mode for a VLAN.
- `vtss_erps_vlan_member_set()` is used to set the ERPS member mode for a VLAN.
- `vtss_erps_port_state_get()` is used to get the forwarding state for an ERPS instance and port.
- `vtss_erps_port_state_set()` is used to set the forwarding state for an ERPS instance and port.

By default, all VLANs are disabled for all rings and all ports are discarding for all rings.

2.20 VStaX

The VStaX functions are used to setup stacking.

- `vtss_vstax_conf_get()` is used to get the VStaX configuration for the switch.
- `vtss_vstax_conf_set()` is used to set the VStaX configuration for the switch.
- `vtss_vstax_port_conf_get()` is used to get the VStaX port configuration.
- `vtss_vstax_port_conf_set()` is used to set the VStaX port configuration.
- `vtss_vstax_master_upsid_get()` is used to get the UPSID of the stack master.
- `vtss_vstax_master_upsid_set()` is used to set the UPSID of the stack master.
- `vtss_vstax_topology_set()` is used to set the stack topology table.

By default, VStaX is disabled for all ports.

Chapter 3

Layer 3

The Layer 3 functionality is build around some global configuration and three tables: router-legs (RLEG), longest-prefix-match (LPM) and neighbour-table (ARP). Following is the purpose of these configurations entities.

Configuration entities

Global configuration: Required to enable routing and to configure the MAC addressing schema to use for the router legs.

Router legs: The purpose of L3 routing is to forward IP frames from one L3 interface to another. The router legs represents these L3 interfaces. A router leg is associated to a L2 VLAN, it has a MAC-address and a set of other attributes to enable/disable IPv4/IPv6 routing and VRRP settings. If a IP frame is received on a VLAN which has an associated router leg, and the destination MAC of the frame matches the MAC address of the router leg, then the frame is candidate for being routed.

NOTE: The router leg table should be synchronized with the interface table in the operating systems IP stack. When a L3 interface is added in the operating system, a corresponding router leg should be created. When an IP address is assigned to a given L3 interface in the IP stack, then the corresponding interface route should be installed in the LPM table.

Longest prefix match: This is where the actual routing takes place, if a IP frame qualifies for routing then the destination IP address is matched against the LPM table. The best match (the longest prefix match) in the LPM table is used to route the frame. If no match is found in the LPM table, then the frame is forwarded to the CPU. The LPM table should include network routes, host routes and interface routes. An interface route is a route representing the configured IP range of the interface, and with the destination configured to zero.

NOTE: An LPM entry can only be routed in HW if the destination of the route is installed in the neighbour table. If this is not the case the frame will be forwarded to the CPU, which should perform the ARP/NDP resolution route the packet in SW and install the neighbour in HW for future use.

Neighbour table: This table is used to map IP address to (MAC addresses, VLAN). When a packet is being routed in the LPM table, the output is a "next-hop" IP address, this "next-hop" IP address must be translated to a MAC-address and a VLAN before it can be L2 forwarded. If a LPM entry does not have a corresponding neighbour entry, then the packet will be forwarded to the CPU.

The neighbour table should be synchronized with the ARP/NDP table of the operating systems IP stack.

Mode of operation

When a frame is being routed and the best match in the LPM table is a interface route, then the packet will be forwarded to the CPU. The CPU should now start the ARP/NDP protocols in order to figure out what the MAC address is of the destination host. The address resolution has completed the frame can be routed. In order to enable hardware routing for directly connected routes, the derived destination host must be installed as a host route in the LPM table, and an associated neighbour entry should be installed.

When a new network route is installed, the MAC address of the next hop route may not be known in advance. If this is the case the first hit on that route will also cause the frame to be forwarded to the CPU, and the CPU may complete the neighbour discovery and install a neighbour entry. After the neighbour entry has been installed the routing will be performed in HW.

Chapter 4

Security

The Security functions are used to control Port Authentication and Access Control List.

4.1 Port Authentication (802.1X)

The the 802.1X forwarding state for a port can be setup.

- [vtss_auth_port_state_get\(\)](#) is used to get the authentication state for a port.
- [vtss_auth_port_state_set\(\)](#) is used to set the authentication state for a port.

By default, all ports are authenticated in both directions.

4.2 Access Control List

Advanced frame processing can be setup using the Access Control List (ACL) of the switch. Each rule in the list is called an Access Control Entry (ACE). For each frame received on an ingress port, the ACL is searched until an ACE matching the ingress port and frame properties is found. The action of the first matching ACE determines the forwarding of the frame. If no matching ACE is found, the default action of the ingress port is used.

By default, the ACL is empty and the default port actions allow forwarding of frames.

4.2.1 Access Control Entry

Each ACE is idenfied by an ACE ID used for identification and ordering of ACEs. The following ACE functions are available:

- [vtss_ace_init\(\)](#) is used to initialize an ACE to default values.
- [vtss_ace_add\(\)](#) is used to add or modify an ACE.
- [vtss_ace_del\(\)](#) is used to delete an ACE.
- [vtss_ace_counter_get\(\)](#) is used to get the hit counter of an ACE.

- `vtss_ace_counter_clear()` is used to clear the hit counter of an ACE.

The `vtss_ace_t` structure used when adding an ACE can be divided into three parts:

- ACE ID (`vtss_ace_id_t`) used for identifying the rule.
- ACE action field (`vtss_acl_action_t`) used to describe the forwarding of matching frames, including:
 - Filtering (e.g. permit/deny frames)
 - Policing (rate limiting)
 - CPU copy (e.g. for protocol processing)
- ACE key fields used to match against incoming frames, including:
 - Ingress ports
 - ACL policy number
 - Frame type and frame type specific fields

4.2.2 Port Configuration

The following ACL functions are available per ingress port:

- `vtss_acl_port_conf_get()` is used to get the ACL port configuration.
- `vtss_acl_port_conf_set()` is used to set the ACL port configuration.
- `vtss_acl_port_counter_get()` is used to get the default hit counter.
- `vtss_acl_port_counter_clear()` is used to clear the default hit counter.

The port configuration includes the default ACL action and the ACL policy number (`vtss_acl_policy_no_t`) which can be used to form groups of ports matching the same ACEs. ACEs matching an ACL policy number can be added to match frames from ports with the same policy number. The value `VTSS_ACL_POLICY_NO_NONE` is used to disable ACL processing on a port.

4.2.3 Policer Configuration

Each policer is identified by an ACL policer ID (`vtss_acl_policer_no_t`) which can be used when mapping an ACL action to a policer. The following functions are available per ACL policer:

- `vtss_acl_policer_conf_get()` is used to get the ACL policer configuration.
- `vtss_acl_policer_conf_set()` is used to set the ACL policer configuration.

Chapter 5

Data Structure Index

5.1 Data Structures

Here are the data structures with brief descriptions:

ib_par_cfg	Generalized data structure for IB parameters	31
port_custom_conf_t	Port configuration	32
serdes_fields_t	Serdes fields	35
vtss_ace_frame_arp_t	Frame data for VTSS_ACE_TYPE_ARP	36
vtss_ace_frame_etype_t	Frame data for VTSS_ACE_TYPE_ETYPE	39
vtss_ace_frame_ipv4_t	Frame data for VTSS_ACE_TYPE_IPV4	40
vtss_ace_frame_ipv6_t	Frame data for VTSS_ACE_TYPE_IPV6	45
vtss_ace_frame_llc_t	Frame data for VTSS_ACE_TYPE_LLCC	49
vtss_ace_frame_snap_t	Frame data for VTSS_ACE_TYPE_SNAP	50
vtss_ace_t	Access Control Entry	51
vtss_ace_vlan_t	ACE VLAN information	55
vtss_acl_action_t	ACL Action	56
vtss_acl_policer_conf_t	ACL policer configuration	59
vtss_acl_port_conf_t	ACL port configuration	60
vtss_aggr_mode_t	Aggregation traffic distribution mode	61
vtss_aneg_t	Auto negotiation struct	62
vtss_api_lock_t	API lock structure	63
vtss_basic_counters_t	Basic counters structure	65

<code>vtss_chip_id_t</code>	Chip ID	65
<code>vtss_counter_pair_t</code>	Counter pair	66
<code>vtss_debug_info_t</code>	Debug information structure	67
<code>vtss_debug_lock_t</code>	API debug lock structure	69
<code>vtss_dgroup_port_conf_t</code>	Destination group port configuration	70
<code>vtss_dlb_policer_conf_t</code>	Dual leaky buckets policer configuration	71
<code>vtss_ece_action_t</code>	ECE action	73
<code>vtss_ece_frame_ipv4_t</code>	ECE IPv4 information	75
<code>vtss_ece_frame_ipv6_t</code>	ECE IPv6 information	76
<code>vtss_ece_inner_tag_t</code>	ECE inner tag	77
<code>vtss_ece_key_t</code>	ECE key	78
<code>vtss_ece_mac_t</code>	ECE MAC information	80
<code>vtss_ece_outer_tag_t</code>	ECE outer tag	81
<code>vtss_ece_t</code>	EVC Control Entry	83
<code>vtss_ece_tag_t</code>	ECE tag information	84
<code>vtss_eee_port_conf_t</code>	EEE port configuration	86
<code>vtss_eee_port_counter_t</code>	EEE port counters (JR only)	87
<code>vtss_eee_port_state_t</code>	EEE port state (JR only)	89
<code>vtss_eps_port_conf_t</code>	Port protection configuration	90
<code>vtss_evc_conf_t</code>	EVC configuration (excluding UNIs)	91
<code>vtss_evc_counters_t</code>	EVC/ECE counters	92
<code>vtss_evc_pb_conf_t</code>	PB specific EVC configuration	94
<code>vtss_evc_port_conf_t</code>	EVC port configuration	96
<code>vtss_ewis_aisl_cons_act_s</code>	EWIS AIS-L consequent actions	97
<code>vtss_ewis_conf_s</code>	EWIS configuration primitives, used to hold all the configuration parameters in the internal state in the API	98
<code>vtss_ewis_cons_act_s</code>	EWIS consequent actions	101
<code>vtss_ewis_counter_s</code>	EWIS performance counters. These counters are free running counters that wraps to zero	102
<code>vtss_ewis_counter_threshold_s</code>	EWIS performance counter thresholds	104

vtss_ewis_defects_s	EWIS defects	105
vtss_ewis_fault_cons_act_s	EWIS fault mask configuration, i.e set up which defects trigger the Fault condition	109
vtss_ewis_force_mode_s	EWIS force modes	112
vtss_ewis_line_force_mode_s	EWIS line force mode	113
vtss_ewis_line_tx_force_mode_s	EWIS line TX force mode	114
vtss_ewis_path_force_mode_s	EWIS path force modes	115
vtss_ewis_perf_mode_s	EWIS Mode(Bit/Block) for the Performance Monitoring Counters	116
vtss_ewis_perf_s	EWIS performance primitives. These data are assumed to be read once every sec. The counters holds increments compared to previous read. The namings and definitions are taken from ITU-T rec G.783	118
vtss_ewis_rdii_cons_act_s	EWIS RDI-L consequent actions	119
vtss_ewis_sl_conf_s	Signal label configuration	121
vtss_ewis_static_conf_s	EWIS static configuration data,	121
vtss_ewis_status_s	EWIS status	125
vtss_ewis_test_conf_s	EWIS test configuration	126
vtss_ewis_test_status_s	EWIS test status	127
vtss_ewis_tti_s	Trail Trace Identifier type	128
vtss_ewis_tx_oh_s	WIS transmitted overhead data. only a few oh bytes can be set dynamically. These OH bytes are not configurable from the API: H4 : multiframe indicator M0/M1: STS-1 Line Remote Error Indication (REI) G1: Path status	129
vtss_ewis_tx_passthru_s	EWIS overhead passthru configuration	133
vtss_fan_conf_t	Fan specifications	137
vtss_gpio_10g_gpio_mode_t	GPIO configured mode	138
vtss_init_conf_t	Initialization configuration	141
vtss_inst_create_t	Create structure	145
vtss_ip_addr_t	Either an IPv4 or IPv6 address	146
vtss_ip_network_t	IPv6 network	147
vtss_ipv4_network_t	IPv4 network	148
vtss_ipv4_uc_t	IPv4 unicast routing entry	149
vtss_ipv6_network_t	IPv6 network	150
vtss_ipv6_t	IPv6 address/mask	151

vtss_ipv6_uc_t	IPv6 routing entry	152
vtss_irq_conf_t	Interrupt configuration options	153
vtss_irq_status_t	Interrupt status structure	154
vtss_l3_common_conf_t	Common configurations for all routing legs	155
vtss_l3_counters_t	Routing interface statics counter	156
vtss_l3_neighbour_t	Neighbour entry	158
vtss_l3_rleg_conf_t	Router leg control structure	160
vtss_lcpll_status_t	Structure for Get PHY LC-PLL status	162
vtss_learn_mode_t	Learning mode	164
vtss_mac_t	MAC Address	165
vtss_mac_table_entry_t	MAC address entry	166
vtss_mac_table_status_t	MAC address table status	169
vtss_mirror_conf_t	Mirror configuration	170
vtss_mtimer_t	Timer structure	172
vtss_npi_conf_t	NPI configuration	173
vtss_os_timestamp_t	174
vtss_packet_dma_conf_t	175
vtss_packet_frame_info_t	Information about frame	176
vtss_packet_port_filter_t	Packet information for each port	177
vtss_packet_port_info_t	Port info structure	178
vtss_packet_rx_conf_t	CPU Rx configuration	180
vtss_packet_rx_header_t	System frame header describing received frame	181
vtss_packet_rx_info_t	Decoded extraction header properties	183
vtss_packet_rx_meta_t	Input structure to <code>vtss_packet_rx_hdr_decode()</code>	193
vtss_packet_rx_port_conf_t	Packet registration per port	197
vtss_packet_rx_queue_conf_t	CPU Rx queue configuration	198
vtss_packet_rx_queue_map_t	CPU Rx queue map	199
vtss_packet_rx_queue_npi_conf_t	CPU Rx queue NPI configuration	202
vtss_packet_rx_reg_t	CPU Rx packet registration	203
vtss_packet_tx_ifh_t	Compiled Tx Frame Header	204

vtss_packet_tx_info_t	Injection Properties	205
vtss_phy_10g_apc_conf_t	10G Phy APC configuration	216
vtss_phy_10g_apc_status_t	10G Phy APC status	217
vtss_phy_10g_auto_failover_conf_t	10G PHY Automatic Failover configuration	218
vtss_phy_10g_base_kr_autoneg_t	10G Phy Base KR Autoneg config	221
vtss_phy_10g_base_kr_conf_t	10G Phy 10f_base_kr_conf config data according to 802.3-2008 clause 72.7 Figure 72-11	222
vtss_phy_10g_base_kr_id_adv_abil_t	10G Phy Base Link Advertisement capability config	224
vtss_phy_10g_base_kr_status_t	10G Phy Base KR Training & Autoneg status	225
vtss_phy_10g_base_kr_train_aneg_t	10G Phy Base KR Training & Autoneg config	226
vtss_phy_10g_base_kr_training_t	10G Phy Base KR Training config	228
vtss_phy_10g_ckout_conf_t	10G Phy CKOUT config data	229
vtss_phy_10g_clause_37_adv_t	Advertisement control data for Clause 37 aneg	231
vtss_phy_10g_clause_37_cmn_status_t	Clause 37 Auto-negotiation status for line and host	233
vtss_phy_10g_clause_37_control_t	Clause 37 control struct	234
vtss_phy_10g_clause_37_status_t	Clause 37 Auto-negotiation status	236
vtss_phy_10g_clk_src_t	10G Phy CLOCK Source Selection	237
vtss_phy_10g_cnt_t	10G Phy Sublayer counters	238
vtss_phy_10g_fifo_sync_t	10G OOS workaround options	239
vtss_phy_10g_fw_status_t	Firmware status	240
vtss_phy_10g_host_clk_conf_t	10G Phy Host clock config data	241
vtss_phy_10g_ib_conf_t	10G Phy IB configuration	242
vtss_phy_10g_ib_status_t	10G Phy IB configuration	247
vtss_phy_10g_ib_storage_t	VSCOPE fast scan storage	248
vtss_phy_10g_id_t	10G Phy part number and revision	249
vtss_phy_10g_init_parm_t	10G Phy Initialization configuration	251
vtss_phy_10g_jitter_conf_t	10G Phy Optimisation of jitter performance	252
vtss_phy_10g_kr_status_aneg_t	10G Phy Base KR Autoneg status	253
vtss_phy_10g_kr_status_fec_t	10G Phy Base KR FEC status	255
vtss_phy_10g_kr_status_train_t	10G Phy Base KR Training status	256

vtss_phy_10g_lane_sync_conf_t	10G Phy Lane SYNC Configuration	258
vtss_phy_10g_line_clk_conf_t	10G Phy Line clock config data	259
vtss_phy_10g_loopback_t	10G Phy system and network loopbacks	260
vtss_phy_10g_mode_t	10G Phy operating mode	261
vtss_phy_10g_ob_status_t	10G Phy OB status	274
vtss_phy_10g_pcs_prbs_gen_conf_t	276
vtss_phy_10g_pcs_prbs_mon_conf_t	277
vtss_phy_10g_pkt_gen_conf_t	10G PHY Packet generator configuration	278
vtss_phy_10g_pkt_mon_conf_t	10G PHY Packet Monitor configuration	281
vtss_phy_10g_polarity_inv_t	10G Phy Polarity inversion	283
vtss_phy_10g_prbs_gen_conf_t	285
vtss_phy_10g_prbs_mon_conf_t	10G PHY prbs monitor Configuration	286
vtss_phy_10g_rxckout_conf_t	10G Phy RXCKOUT config data	291
vtss_phy_10g_sckout_conf_t	10G Phy SCKOUT config data	292
vtss_phy_10g_serdes_status_t	10G Phy SERDES status	294
vtss_phy_10g_srefclk_mode_t	10G Phy srefclk config data	301
vtss_phy_10g_status_t	10G Phy link and fault status for all sublayers	301
vtss_phy_10g_timestamp_val_t	10G PHY timestamp value array(holder)	304
vtss_phy_10g_txckout_conf_t	10G Phy TXCKOUT config data	305
vtss_phy_10g_vscope_conf_t	306
vtss_phy_10g_vscope_scan_conf_t	VSCOPE scan configuration	307
vtss_phy_10g_vscope_scan_status_t	309
vtss_phy_aneg_t	PHY auto negotiation advertisement	311
vtss_phy_clock_conf_t	PHY clock configuration	313
vtss_phy_conf_1g_t	PHY 1G configuration	314
vtss_phy_conf_t	PHY configuration	315
vtss_phy_daisy_chain_conf_t	SPI daisy chain configuration	318
vtss_phy_eee_conf_t	EEE configuration	319
vtss_phy_enhanced_led_control_t	Enhanced LED control	320
vtss_phy_forced_t	PHY forced mode configuration	321
vtss_phy_led_mode_select_t	LED model selection	322

vtss_phy_loopback_t	1G Phy loopbacks	323
vtss_phy_ltc_freq_synth_s	Frequency synthesis pulse configuration	326
vtss_phy_mac_serdes_pcs_cntl_t	PHY MAC SerDes PCS Control, Reg16E3	327
vtss_phy_media_serdes_pcs_cntl_t	PHY MEDIA SerDes PCS Control, Reg23E3	330
vtss_phy_pcs_cnt_t	10G Phy PCS counters	332
vtss_phy_power_conf_t	PHY power configuration	334
vtss_phy_power_status_t	PHY power status	335
vtss_phy_reset_conf_t	PHY reset structure	335
vtss_phy_rgmii_conf_t	PHY RGMII configuration	337
vtss_phy_statistic_t	Phy statistic information	338
vtss_phy_status_1g_t	PHY 1G status	340
vtss_phy_tbi_conf_t	PHY TBI configuration	341
vtss_phy_timestamp_t	PHY timestamp in seconds and nanoseconds (10 bytes Timestamp)	342
vtss_phy_ts_ach_conf_t	Analyzer ACH comparator configuration options	343
vtss_phy_ts_alt_clock_mode_s	Parameter for setting the alternative clock mode	345
vtss_phy_ts_eng_init_conf_t	Defines the basic engine parameters passed to the engine_init_conf_get() function	346
vtss_phy_ts_engine_action_t	Engine Action configuration options	348
vtss_phy_ts_engine_flow_conf_t	Analyzer flow configuration options	350
vtss_phy_ts_eth_conf_t	Analyzer Ethernet comparator configuration options	352
vtss_phy_ts_fifo_conf_t	Defines the params for FIFO SYNC function	358
vtss_phy_ts_fifo_sig_t	Tx TSFIFO entry signature	359
vtss_phy_ts_gen_conf_t	Analyzer Generic data configuration options using IP comparator	361
vtss_phy_ts_generic_action_t	Generic Action configuration option	363
vtss_phy_ts_generic_flow_conf_t	Generic engine flow configuration options	365
vtss_phy_ts_ietf_mpls_ach_oam_conf_t	Analyzer OAM comparator, IETF MPLS ACH OAM Packet format configuration options	366
vtss_phy_ts_init_conf_t	Defines the initial parameters to be passed to init function	367
vtss_phy_ts_ip_conf_t	Analyzer IP comparator configuration options	371
vtss_phy_ts_mpls_conf_t	Analyzer MPLS comparator configuration options	375
vtss_phy_ts_mpls_lvl_rng_t	MPLS level range	379

vtss_phy_ts_nphase_status_t	N-phase status	380
vtss_phy_ts_oam_engine_action_t	OAM Action configuration options	381
vtss_phy_ts_oam_engine_flow_conf_t	OAM engine flow configuration options	383
vtss_phy_ts_pps_config_s	PPS Configuration	384
vtss_phy_ts_ptp_conf_t	Analyzer PTP comparator configuration options	386
vtss_phy_ts_ptp_engine_action_t	Analyzer PTP action configuration options	388
vtss_phy_ts_ptp_engine_flow_conf_t	PTP engine flow configuration options	389
vtss_phy_ts_sertod_conf_t	PHY timestamp in seconds and nanoseconds (10 bytes Timestamp)	391
vtss_phy_ts_stats_t	Timestamping Statistics	392
vtss_phy_ts_y1731_oam_conf_t	Analyzer OAM comparator, Y.1731 OAM Packet format configuration options	395
vtss_phy_type_t	Phy type information	397
vtss_phy_veriphy_result_t	VeriPHY result	399
vtss_phy_wol_conf_t	Structure for Get/Set Wake-On-LAN configuration	400
vtss_pi_conf_t	PI configuration	401
vtss_policer_ext_t	Policer Extensions	402
vtss_policer_t	Policer	406
vtss_port_bridge_counters_t	Port bridge counter structure (RFC 4188)	406
vtss_port_clause_37_adv_t	Advertisement control data for Clause 37 aneg	407
vtss_port_clause_37_control_t	Auto-negotiation control parameter struct	409
vtss_port_conf_t	Port configuration structure	410
vtss_port_counters_t	Port counter structure	414
vtss_port_ethernet_like_counters_t	Ethernet-like Interface counter structure (RFC 3635)	416
vtss_port_flow_control_conf_t	Flow control setup	419
vtss_port_frame_gaps_t	Inter frame gap structure	421
vtss_port_if_group_counters_t	Interfaces Group counter structure (RFC 2863)	422
vtss_port_ifh_t	Port Internal Frame Header structure	425
vtss_port_map_t	Port map structure	426
vtss_port_proprietary_counters_t	Port proprietary counter structure	428
vtss_port_rmon_counters_t	RMON counter structure (RFC 2819)	429

vtss_port_serdes_conf_t	SFI Serdes configuration	436
vtss_port_sgmii_aneg_t	Advertisement control data for SGMII aneg	437
vtss_port_status_t	Port status parameter struct	439
vtss_qce_action_t	QCE action	441
vtss_qce_frame_etype_t	Frame data for VTSS_QCE_TYPE_ETYPE	443
vtss_qce_frame_ipv4_t	Frame data for VTSS_QCE_TYPE_IPV4	444
vtss_qce_frame_ipv6_t	Frame data for VTSS_QCE_TYPE_IPV6	446
vtss_qce_frame_llc_t	Frame data for VTSS_QCE_TYPE_LLCC	447
vtss_qce_frame_snap_t	Frame data for VTSS_QCE_TYPE_SNAP	448
vtss_qce_key_t	QCE key	449
vtss_qce_mac_t	QCE MAC information	451
vtss_qce_t	QoS Control Entry	453
vtss_qce_tag_t	QCE tag information	454
vtss_qos_conf_t	All parameters below are defined per chip	455
vtss_qos_port_conf_t	QoS setup per port	457
vtss_qs_conf_t	Queue System settings	463
vtss_rcpll_status_t	Structure for Get PHY RC-PLL status	465
vtss_red_t	Random Early Detection configuration struct version 1 (per port, per queue)	467
vtss_restart_status_t	Restart status	468
vtss_routing_entry_t	Routing entry	470
vtss_secure_on_passwd_t	Structure for Wake-On-LAN Secure-On Password	471
vtss_serdes_macro_conf_t	Serdes macro configuration	472
vtss_sflow_port_conf_t	SFlow configuration structure	473
vtss_sgpio_conf_t	GPIO configuration for a group	474
vtss_sgpio_port_conf_t	GPIO port configuration	475
vtss_sgpio_port_data_t	GPIO read data for a port	477
vtss_shaper_t	Shaper	477
vtss_sublayer_status_t	10G Phy link and fault status	478

vtss_sync_clock_in_t	Struct containing configuration selecting the recovered input clock port, to be delivered to a selected output clock port	480
vtss_sync_clock_out_t	Struct containing configuration for a recovered clock output port	481
vtss_tci_t	Tag Control Information (according to IEEE 802.1Q)	482
vtss_timeofday_t	Time of day structure	483
vtss_timestamp_t	Time stamp in seconds and nanoseconds	484
vtss_trace_conf_t	Trace group configuration	485
vtss_ts_ext_clock_mode_t	External clock output configuration	486
vtss_ts_id_t	Timestamp identifier	487
vtss_ts_internal_mode_t	Hardware timestamping format mode for internal ports	488
vtss_ts_operation_mode_t	Timestamp operation	489
vtss_ts_timestamp_alloc_t	Timestamp allocation	489
vtss_ts_timestamp_t	Timestamp structure	490
vtss_vcap_ip_t	VCAP IPv4 address value and mask	492
vtss_vcap_u128_t	VCAP 128 bit value and mask	493
vtss_vcap_u16_t	VCAP 16 bit value and mask	494
vtss_vcap_u24_t	VCAP 24 bit value and mask	495
vtss_vcap_u32_t	VCAP 32 bit value and mask	496
vtss_vcap_u40_t	VCAP 40 bit value and mask	497
vtss_vcap_u48_t	VCAP 48 bit value and mask	498
vtss_vcap_u8_t	VCAP 8 bit value and mask	499
vtss_vcap_udp_tcp_t	VCAP UDP/TCP port range	500
vtss_vcap_vid_t	VCAP VLAN ID value and mask	501
vtss_vcap_vr_t	VCAP universal value or range	502
vtss_vce_action_t	VCE Action	504
vtss_vce_frame_etype_t	Frame data for VTSS_VCE_TYPE_ETYPE	505
vtss_vce_frame_ipv4_t	Frame data for VTSS_VCE_TYPE_IPV4	506
vtss_vce_frame_ipv6_t	Frame data for VTSS_VCE_TYPE_IPV6	508
vtss_vce_frame_llc_t	Frame data for VTSS_VCE_TYPE_LLCC	509

vtss_vce_frame_snap_t	Frame data for VTSS_VCE_TYPE_SNAP	510
vtss_vce_key_t	VCE Key	511
vtss_vce_mac_t	VCE MAC header information	513
vtss_vce_t	VLAN Control Entry	514
vtss_vce_tag_t	VCE tag information	515
vtss_vcl_port_conf_t	VCL port configuration	517
vtss_vid_mac_t	MAC Address in specific VLAN	518
vtss_vlan_conf_t	VLAN configuration	519
vtss_vlan_port_conf_t	VLAN port configuration	520
vtss_vlan_tag_t	521
vtss_vlan_trans_grp2vlan_conf_t	VLAN translation group-to-VLAN configuration	523
vtss_vlan_trans_port2grp_conf_t	VLAN translation port-to-group configuration	524
vtss_vlan_vid_conf_t	VLAN ID configuration	525
vtss_vstax_conf_t	VStaX configuration for switch	526
vtss_vstax_glag_entry_t	GLAG info	528
vtss_vstax_port_conf_t	VStaX setup for port	528
vtss_vstax_route_entry_t	UPSID Route Entry	529
vtss_vstax_route_table_t	UPSID Route Table	530
vtss_vstax_rx_header_t	VStaX frame header used for reception	531
vtss_vstax_tx_header_t	VStaX frame header used for transmission	533
vtss_wol_mac_addr_t	Structure for Wake-On-LAN MAC Address	536

Chapter 6

File Index

6.1 File List

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

vtss_api/include/ vtss_ae_api.h	
Ae API	609
vtss_api/include/ vtss_afi_api.h	
AFI API	609
vtss_api/include/ vtss_aneg_api.h	
ANEG API	610
vtss_api/include/ vtss_api.h	
Vitesse API main header file	610
vtss_api/include/ vtss_evc_api.h	
EVC API	610
vtss_api/include/ vtss_fdma_api.h	
Frame DMA API	621
vtss_api/include/ vtss_gfp_api.h	
GFP API	621
vtss_api/include/ vtss_hqos_api.h	
HQoS API	621
vtss_api/include/ vtss_i2c_api.h	
I2C API	622
vtss_api/include/ vtss_init_api.h	
Initialization API	622
vtss_api/include/ vtss_l2_api.h	
Layer 2 API	637
vtss_api/include/ vtss_l3_api.h	
L3 routing API	702
vtss_api/include/ vtss_mac10g_api.h	
MAC10G API	714
vtss_api/include/ vtss_macsec_api.h	
vtss_api/include/ vtss_misc_api.h	
Miscellaneous API	714
vtss_api/include/ vtss_mpls_api.h	
MPLS API	746
vtss_api/include/ vtss_oam_api.h	
OAM API	746
vtss_api/include/ vtss_oha_api.h	
OHA API	747

vtss_api/include/vtss_os.h	
OS Layer API	747
vtss_api/include/vtss_os_custom.h	
OS custom header file	747
vtss_api/include/vtss_os_ecos.h	
ECos OS API	752
vtss_api/include/vtss_os_linux.h	
Linux OS API	762
vtss_api/include/vtss_otn_api.h	
OTN API	769
vtss_api/include/vtss_packet_api.h	
Packet API	769
vtss_api/include/vtss_pcs_10gbase_r_api.h	
PCS_10BASE_R API	792
vtss_api/include/vtss_phy_10g_api.h	
10G PHY API	792
vtss_api/include/vtss_phy_api.h	
PHY API	886
vtss_api/include/vtss_phy_ts_api.h	
PHY TimeStamping API	962
vtss_api/include/vtss_port_api.h	
Port API	1031
vtss_api/include/vtss_qos_api.h	
QoS API	1048
vtss_api/include/vtss_rab_api.h	
RAB API	1057
vtss_api/include/vtss_security_api.h	
Security API	1057
vtss_api/include/vtss_sfi4_api.h	
SFI4 API	1067
vtss_api/include/vtss_sync_api.h	
Synchronization API	1067
vtss_api/include/vtss_tfi5_api.h	
TFI5 API	1071
vtss_api/include/vtss_ts_api.h	
TimeStamping API	1071
vtss_api/include/vtss_upi_api.h	
Define UPI API interface	1088
vtss_api/include/vtss_wis_api.h	
EWIS layer API	1088
vtss_api/include/vtss_xaui_api.h	
XAUI API	1118
vtss_api/include/vtss_xfi_api.h	
XFI API	1118
vtss_api/include/vtss/api/l2_types.h	
Layer 2 Public API Header for l2	537
vtss_api/include/vtss/api/options.h	
Features and options	538
vtss_api/include/vtss/api/phy.h	
PHY Public API Header	556
vtss_api/include/vtss/api/port.h	
Port Public API Header	558
vtss_api/include/vtss/api/types.h	
Generic types API	571

Chapter 7

Data Structure Documentation

7.1 ib_par_cfg Struct Reference

Generalized data structure for IB parameters.

```
#include <vtss_phy_10g_api.h>
```

Data Fields

- u16 value
- u16 min
- u16 max

7.1.1 Detailed Description

Generalized data structure for IB parameters.

Definition at line 207 of file vtss_phy_10g_api.h.

7.1.2 Field Documentation

7.1.2.1 value

```
u16 ib_par_cfg::value
```

value to be configured

Definition at line 208 of file vtss_phy_10g_api.h.

7.1.2.2 min

`u16 ib_par_cfg::min`

Minimum value

Definition at line 209 of file `vtss_phy_10g_api.h`.

7.1.2.3 max

`u16 ib_par_cfg::max`

Maximum value

Definition at line 210 of file `vtss_phy_10g_api.h`.

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

- `vtss_api/include/vtss_phy_10g_api.h`

7.2 port_custom_conf_t Struct Reference

Port configuration.

```
#include <port.h>
```

Data Fields

- `BOOL enable`
- `BOOL autoneg`
- `BOOL fdx`
- `BOOL flow_control`
- `BOOL pfc [VTSS_PRIOS]`
- `vtss_port_speed_t speed`
- `vtss_fiber_port_speed_t dual_media_fiber_speed`
- `unsigned int max_length`
- `BOOL exc_col_cont`
- `u8 adv_dis`
- `u8 max_tags`
- `BOOL oper_up`
- `BOOL frame_length_chk`

7.2.1 Detailed Description

Port configuration.

Definition at line 269 of file `port.h`.

7.2.2 Field Documentation

7.2.2.1 enable

`BOOL port_custom_conf_t::enable`

Admin enable/disable

Definition at line 270 of file port.h.

7.2.2.2 autoneg

`BOOL port_custom_conf_t::autoneg`

Auto negotiation

Definition at line 271 of file port.h.

7.2.2.3 fdx

`BOOL port_custom_conf_t::fdx`

Forced duplex mode

Definition at line 272 of file port.h.

7.2.2.4 flow_control

`BOOL port_custom_conf_t::flow_control`

Flow control (Standard 802.3x)

Definition at line 273 of file port.h.

7.2.2.5 pfc

`BOOL port_custom_conf_t::pfc[VTSS_PRIOS]`

Priority Flow control (802.1Qbb)

Definition at line 275 of file port.h.

7.2.2.6 speed

```
vtss_port_speed_t port_custom_conf_t::speed
```

Forced port speed

Definition at line 277 of file port.h.

7.2.2.7 dual_media_fiber_speed

```
vtss_fiber_port_speed_t port_custom_conf_t::dual_media_fiber_speed
```

Speed for dual media fiber ports

Definition at line 278 of file port.h.

7.2.2.8 max_length

```
unsigned int port_custom_conf_t::max_length
```

Max frame length

Definition at line 279 of file port.h.

7.2.2.9 exc_col_cont

```
BOOL port_custom_conf_t::exc_col_cont
```

Excessive collision continuation

Definition at line 283 of file port.h.

7.2.2.10 adv_dis

```
u8 port_custom_conf_t::adv_dis
```

Auto neg advertisement disable

Definition at line 284 of file port.h.

7.2.2.11 max_tags

`u8 port_custom_conf_t::max_tags`

Maximum number of tags

Definition at line 285 of file port.h.

7.2.2.12 oper_up

`BOOL port_custom_conf_t::oper_up`

Force operational state up

Definition at line 286 of file port.h.

7.2.2.13 frame_length_chk

`BOOL port_custom_conf_t::frame_length_chk`

True to do 802.3 frame length check for ethertypes below 0x0600

Definition at line 287 of file port.h.

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

- `vtss_api/include/vtss/api/port.h`

7.3 serdes_fields_t Struct Reference

Serdes fields.

```
#include <vtss_init_api.h>
```

Data Fields

- `u32 ob_post0`
- `u32 ob_sr`

7.3.1 Detailed Description

Serdes fields.

Definition at line 357 of file vtss_init_api.h.

7.3.2 Field Documentation

7.3.2.1 ob_post0

`u32 serdes_fields_t::ob_post0`

Trace length

Definition at line 358 of file vtss_init_api.h.

7.3.2.2 ob_sr

`u32 serdes_fields_t::ob_sr`

Slew Rate

Definition at line 359 of file vtss_init_api.h.

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

- `vtss_api/include/vtss_init_api.h`

7.4 vtss_ace_frame_arp_t Struct Reference

Frame data for VTSS_ACE_TYPE_ARP.

```
#include <vtss_security_api.h>
```

Data Fields

- `vtss_ace_u48_t smac`
- `vtss_ace_bit_t arp`
- `vtss_ace_bit_t req`
- `vtss_ace_bit_t unknown`
- `vtss_ace_bit_t smac_match`
- `vtss_ace_bit_t dmac_match`
- `vtss_ace_bit_t length`
- `vtss_ace_bit_t ip`
- `vtss_ace_bit_t ethernet`
- `vtss_ace_ip_t sip`
- `vtss_ace_ip_t dip`

7.4.1 Detailed Description

Frame data for VTSS_ACE_TYPE_ARP.

Definition at line 450 of file vtss_security_api.h.

7.4.2 Field Documentation

7.4.2.1 smac

`vtss_ace_u48_t vtss_ace_frame_arp_t::smac`

SMAC

Definition at line 452 of file vtss_security_api.h.

7.4.2.2 arp

`vtss_ace_bit_t vtss_ace_frame_arp_t::arp`

Opcode ARP/RARP

Definition at line 453 of file vtss_security_api.h.

7.4.2.3 req

`vtss_ace_bit_t vtss_ace_frame_arp_t::req`

Opcode request/reply

Definition at line 454 of file vtss_security_api.h.

7.4.2.4 unknown

`vtss_ace_bit_t vtss_ace_frame_arp_t::unknown`

Opcode unknown

Definition at line 455 of file vtss_security_api.h.

7.4.2.5 smac_match

`vtss_ace_bit_t vtss_ace_frame_arp_t::smac_match`

Sender MAC matches SMAC

Definition at line 456 of file vtss_security_api.h.

7.4.2.6 dmac_match

`vtss_ace_bit_t vtss_ace_frame_arp_t::dmac_match`

Target MAC matches DMAC

Definition at line 457 of file vtss_security_api.h.

7.4.2.7 length

`vtss_ace_bit_t vtss_ace_frame_arp_t::length`

Protocol addr. length 4, hardware length 6

Definition at line 458 of file vtss_security_api.h.

7.4.2.8 ip

`vtss_ace_bit_t vtss_ace_frame_arp_t::ip`

Protocol address type IP

Definition at line 459 of file vtss_security_api.h.

7.4.2.9 ethernet

`vtss_ace_bit_t vtss_ace_frame_arp_t::ethernet`

Hardware address type Ethernet

Definition at line 460 of file vtss_security_api.h.

7.4.2.10 sip

`vtss_ace_ip_t` `vtss_ace_frame_arp_t::sip`

Sender IP address

Definition at line 461 of file `vtss_security_api.h`.

7.4.2.11 dip

`vtss_ace_ip_t` `vtss_ace_frame_arp_t::dip`

Target IP address

Definition at line 462 of file `vtss_security_api.h`.

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

- `vtss_api/include/vtss_security_api.h`

7.5 vtss_ace_frame_etype_t Struct Reference

Frame data for `VTSS_ACE_TYPE_ETYPE`.

```
#include <vtss_security_api.h>
```

Data Fields

- `vtss_ace_u48_t dmac`
- `vtss_ace_u48_t smac`
- `vtss_ace_u16_t etype`
- `vtss_ace_u16_t data`

7.5.1 Detailed Description

Frame data for `VTSS_ACE_TYPE_ETYPE`.

Definition at line 422 of file `vtss_security_api.h`.

7.5.2 Field Documentation

7.5.2.1 dmac

`vtss_ace_u48_t vtss_ace_frame_etype_t::dmac`

DMAC

Definition at line 424 of file `vtss_security_api.h`.

7.5.2.2 smac

`vtss_ace_u48_t vtss_ace_frame_etype_t::smac`

SMAC

Definition at line 425 of file `vtss_security_api.h`.

7.5.2.3 etype

`vtss_ace_u16_t vtss_ace_frame_etype_t::etype`

Ethernet Type value

Definition at line 426 of file `vtss_security_api.h`.

7.5.2.4 data

`vtss_ace_u16_t vtss_ace_frame_etype_t::data`

MAC data

Definition at line 427 of file `vtss_security_api.h`.

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

- `vtss_api/include/vtss_security_api.h`

7.6 `vtss_ace_frame_ipv4_t` Struct Reference

Frame data for VTSS_ACE_TYPE_IPV4.

```
#include <vtss_security_api.h>
```

Data Fields

- `vtss_ace_bit_t ttl`
- `vtss_ace_bit_t fragment`
- `vtss_ace_bit_t options`
- `vtss_ace_u8_t ds`
- `vtss_ace_u8_t proto`
- `vtss_ace_ip_t sip`
- `vtss_ace_ip_t dip`
- `vtss_ace_u48_t data`
- `vtss_ace_udp_tcp_t sport`
- `vtss_ace_udp_tcp_t dport`
- `vtss_ace_bit_t tcp_fin`
- `vtss_ace_bit_t tcp_syn`
- `vtss_ace_bit_t tcp_rst`
- `vtss_ace_bit_t tcp_psh`
- `vtss_ace_bit_t tcp_ack`
- `vtss_ace_bit_t tcp_urg`
- `vtss_ace_bit_t sip_eq_dip`
- `vtss_ace_bit_t sport_eq_dport`
- `vtss_ace_bit_t seq_zero`

7.6.1 Detailed Description

Frame data for VTSS_ACE_TYPE_IPV4.

Definition at line 466 of file vtss_security_api.h.

7.6.2 Field Documentation

7.6.2.1 ttl

`vtss_ace_bit_t vtss_ace_frame_ipv4_t::ttl`

TTL zero

Definition at line 468 of file vtss_security_api.h.

7.6.2.2 fragment

`vtss_ace_bit_t vtss_ace_frame_ipv4_t::fragment`

Fragment

Definition at line 469 of file vtss_security_api.h.

7.6.2.3 options

`vtss_ace_bit_t` `vtss_ace_frame_ipv4_t::options`

Header options

Definition at line 470 of file vtss_security_api.h.

7.6.2.4 ds

`vtss_ace_u8_t` `vtss_ace_frame_ipv4_t::ds`

DS field

Definition at line 471 of file vtss_security_api.h.

7.6.2.5 proto

`vtss_ace_u8_t` `vtss_ace_frame_ipv4_t::proto`

Protocol

Definition at line 472 of file vtss_security_api.h.

7.6.2.6 sip

`vtss_ace_ip_t` `vtss_ace_frame_ipv4_t::sip`

Source IP address

Definition at line 473 of file vtss_security_api.h.

7.6.2.7 dip

`vtss_ace_ip_t` `vtss_ace_frame_ipv4_t::dip`

Destination IP address

Definition at line 474 of file vtss_security_api.h.

7.6.2.8 data

`vtss_ace_u48_t vtss_ace_frame_ipv4_t::data`

Not UDP/TCP: IP data

Definition at line 475 of file vtss_security_api.h.

7.6.2.9 sport

`vtss_ace_udp_tcp_t vtss_ace_frame_ipv4_t::sport`

UDP/TCP: Source port

Definition at line 476 of file vtss_security_api.h.

7.6.2.10 dport

`vtss_ace_udp_tcp_t vtss_ace_frame_ipv4_t::dport`

UDP/TCP: Destination port

Definition at line 477 of file vtss_security_api.h.

7.6.2.11 tcp_fin

`vtss_ace_bit_t vtss_ace_frame_ipv4_t::tcp_fin`

TCP FIN

Definition at line 478 of file vtss_security_api.h.

7.6.2.12 tcp_syn

`vtss_ace_bit_t vtss_ace_frame_ipv4_t::tcp_syn`

TCP SYN

Definition at line 479 of file vtss_security_api.h.

7.6.2.13 tcp_rst

`vtss_ace_bit_t vtss_ace_frame_ipv4_t::tcp_rst`

TCP RST

Definition at line 480 of file vtss_security_api.h.

7.6.2.14 tcp_psh

`vtss_ace_bit_t vtss_ace_frame_ipv4_t::tcp_psh`

TCP PSH

Definition at line 481 of file vtss_security_api.h.

7.6.2.15 tcp_ack

`vtss_ace_bit_t vtss_ace_frame_ipv4_t::tcp_ack`

TCP ACK

Definition at line 482 of file vtss_security_api.h.

7.6.2.16 tcp_urg

`vtss_ace_bit_t vtss_ace_frame_ipv4_t::tcp_urg`

TCP URG

Definition at line 483 of file vtss_security_api.h.

7.6.2.17 sip_eq_dip

`vtss_ace_bit_t vtss_ace_frame_ipv4_t::sip_eq_dip`

SIP equals DIP

Definition at line 484 of file vtss_security_api.h.

7.6.2.18 sport_eq_dport

`vtss_ace_bit_t vtss_ace_frame_ipv4_t::sport_eq_dport`

SPORT equals DPORt

Definition at line 485 of file vtss_security_api.h.

7.6.2.19 seq_zero

`vtss_ace_bit_t vtss_ace_frame_ipv4_t::seq_zero`

TCP sequence number is zero

Definition at line 486 of file vtss_security_api.h.

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

- vtss_api/include/[vtss_security_api.h](#)

7.7 vtss_ace_frame_ipv6_t Struct Reference

Frame data for VTSS_ACE_TYPE_IPV6.

```
#include <vtss_security_api.h>
```

Data Fields

- [vtss_ace_u8_t proto](#)
- [vtss_ace_u128_t sip](#)
- [vtss_ace_bit_t ttl](#)
- [vtss_ace_u8_t ds](#)
- [vtss_ace_u48_t data](#)
- [vtss_ace_udp_tcp_t sport](#)
- [vtss_ace_udp_tcp_t dport](#)
- [vtss_ace_bit_t tcp_fin](#)
- [vtss_ace_bit_t tcp_syn](#)
- [vtss_ace_bit_t tcp_rst](#)
- [vtss_ace_bit_t tcp_psh](#)
- [vtss_ace_bit_t tcp_ack](#)
- [vtss_ace_bit_t tcp_urg](#)
- [vtss_ace_bit_t sip_eq_dip](#)
- [vtss_ace_bit_t sport_eq_dport](#)
- [vtss_ace_bit_t seq_zero](#)

7.7.1 Detailed Description

Frame data for VTSS_ACE_TYPE_IPV6.

Definition at line 494 of file vtss_security_api.h.

7.7.2 Field Documentation

7.7.2.1 proto

`vtss_ace_u8_t` `vtss_ace_frame_ipv6_t::proto`

IPv6 protocol

Definition at line 496 of file vtss_security_api.h.

7.7.2.2 sip

`vtss_ace_u128_t` `vtss_ace_frame_ipv6_t::sip`

IPv6 source address (byte 0-7 ignored for ACL_V2)

Definition at line 497 of file vtss_security_api.h.

7.7.2.3 ttl

`vtss_ace_bit_t` `vtss_ace_frame_ipv6_t::ttl`

TTL zero

Definition at line 498 of file vtss_security_api.h.

7.7.2.4 ds

`vtss_ace_u8_t` `vtss_ace_frame_ipv6_t::ds`

DS field

Definition at line 499 of file vtss_security_api.h.

7.7.2.5 data

`vtss_ace_u48_t vtss_ace_frame_ipv6_t::data`

Not UDP/TCP: IP data

Definition at line 500 of file vtss_security_api.h.

7.7.2.6 sport

`vtss_ace_udp_tcp_t vtss_ace_frame_ipv6_t::sport`

UDP/TCP: Source port

Definition at line 501 of file vtss_security_api.h.

7.7.2.7 dport

`vtss_ace_udp_tcp_t vtss_ace_frame_ipv6_t::dport`

UDP/TCP: Destination port

Definition at line 502 of file vtss_security_api.h.

7.7.2.8 tcp_fin

`vtss_ace_bit_t vtss_ace_frame_ipv6_t::tcp_fin`

TCP FIN

Definition at line 503 of file vtss_security_api.h.

7.7.2.9 tcp_syn

`vtss_ace_bit_t vtss_ace_frame_ipv6_t::tcp_syn`

TCP SYN

Definition at line 504 of file vtss_security_api.h.

7.7.2.10 `tcp_rst`

`vtss_ace_bit_t` `vtss_ace_frame_ipv6_t::tcp_rst`

TCP RST

Definition at line 505 of file `vtss_security_api.h`.

7.7.2.11 `tcp_psh`

`vtss_ace_bit_t` `vtss_ace_frame_ipv6_t::tcp_psh`

TCP PSH

Definition at line 506 of file `vtss_security_api.h`.

7.7.2.12 `tcp_ack`

`vtss_ace_bit_t` `vtss_ace_frame_ipv6_t::tcp_ack`

TCP ACK

Definition at line 507 of file `vtss_security_api.h`.

7.7.2.13 `tcp_urg`

`vtss_ace_bit_t` `vtss_ace_frame_ipv6_t::tcp_urg`

TCP URG

Definition at line 508 of file `vtss_security_api.h`.

7.7.2.14 `sip_eq_dip`

`vtss_ace_bit_t` `vtss_ace_frame_ipv6_t::sip_eq_dip`

SIP equals DIP

Definition at line 509 of file `vtss_security_api.h`.

7.7.2.15 sport_eq_dport

`vtss_ace_bit_t` `vtss_ace_frame_ipv6_t::sport_eq_dport`

SPORT equals DPORt

Definition at line 510 of file `vtss_security_api.h`.

7.7.2.16 seq_zero

`vtss_ace_bit_t` `vtss_ace_frame_ipv6_t::seq_zero`

TCP sequence number is zero

Definition at line 511 of file `vtss_security_api.h`.

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

- `vtss_api/include/vtss_security_api.h`

7.8 vtss_ace_frame_llc_t Struct Reference

Frame data for VTSS_ACE_TYPE LLC.

```
#include <vtss_security_api.h>
```

Data Fields

- `vtss_ace_u48_t dmac`
- `vtss_ace_u48_t smac`
- `vtss_ace_u32_t llc`

7.8.1 Detailed Description

Frame data for VTSS_ACE_TYPE LLC.

Definition at line 434 of file `vtss_security_api.h`.

7.8.2 Field Documentation

7.8.2.1 dmac

`vtss_ace_u48_t vtss_ace_frame_llc_t::dmac`

DMAC

Definition at line 436 of file `vtss_security_api.h`.

7.8.2.2 smac

`vtss_ace_u48_t vtss_ace_frame_llc_t::smac`

SMAC

Definition at line 437 of file `vtss_security_api.h`.

7.8.2.3 llc

`vtss_ace_u32_t vtss_ace_frame_llc_t::llc`

LLC header: DSAP at byte 0, SSAP at byte 1, Control at byte 2

Definition at line 438 of file `vtss_security_api.h`.

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

- `vtss_api/include/vtss_security_api.h`

7.9 `vtss_ace_frame_snap_t` Struct Reference

Frame data for VTSS_ACE_TYPE_SNAP.

```
#include <vtss_security_api.h>
```

Data Fields

- `vtss_ace_u48_t dmac`
- `vtss_ace_u48_t smac`
- `vtss_ace_u40_t snap`

7.9.1 Detailed Description

Frame data for VTSS_ACE_TYPE_SNAP.

Definition at line 442 of file `vtss_security_api.h`.

7.9.2 Field Documentation

7.9.2.1 dmac

`vtss_ace_u48_t vtss_ace_frame_snap_t::dmac`

DMAC

Definition at line 444 of file `vtss_security_api.h`.

7.9.2.2 smac

`vtss_ace_u48_t vtss_ace_frame_snap_t::smac`

SMAC

Definition at line 445 of file `vtss_security_api.h`.

7.9.2.3 snap

`vtss_ace_u40_t vtss_ace_frame_snap_t::snap`

SNAP header: Organization Code at byte 0, Type at byte 3

Definition at line 446 of file `vtss_security_api.h`.

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

- `vtss_api/include/vtss_security_api.h`

7.10 vtss_ace_t Struct Reference

Access Control Entry.

```
#include <vtss_security_api.h>
```

Data Fields

- `vtss_ace_id_t id`
- `vtss_port_no_t port_no`
- `vtss_ace_u8_t policy`
- `vtss_ace_type_t type`
- `vtss_acl_action_t action`
- `vtss_ace_bit_t dmac_mc`
- `vtss_ace_bit_t dmac_bc`
- `vtss_ace_vlan_t vlan`
- union {
 - `vtss_ace_frame_etype_t etype`
 - `vtss_ace_frame_llc_t llc`
 - `vtss_ace_frame_snap_t snap`
 - `vtss_ace_frame_arp_t arp`
 - `vtss_ace_frame_ipv4_t ipv4`
 - `vtss_ace_frame_ipv6_t ipv6`}
- `frame`

7.10.1 Detailed Description

Access Control Entry.

Definition at line 518 of file `vtss_security_api.h`.

7.10.2 Field Documentation

7.10.2.1 id

`vtss_ace_id_t vtss_ace_t::id`

ACE ID, must be different from `VTSS_ACE_ID_LAST`

Definition at line 520 of file `vtss_security_api.h`.

7.10.2.2 port_no

`vtss_port_no_t vtss_ace_t::port_no`

Port number or `VTSS_PORT_NO_ANY`

Definition at line 527 of file `vtss_security_api.h`.

7.10.2.3 policy

`vtss_ace_u8_t vtss_ace_t::policy`

Policy number

Definition at line 532 of file `vtss_security_api.h`.

7.10.2.4 type

`vtss_ace_type_t vtss_ace_t::type`

ACE frame type

Definition at line 533 of file `vtss_security_api.h`.

7.10.2.5 action

`vtss_acl_action_t vtss_ace_t::action`

ACE action

Definition at line 534 of file `vtss_security_api.h`.

7.10.2.6 dmac_mc

`vtss_ace_bit_t vtss_ace_t::dmac_mc`

Multicast DMAC

Definition at line 536 of file `vtss_security_api.h`.

7.10.2.7 dmac_bc

`vtss_ace_bit_t vtss_ace_t::dmac_bc`

Broadcast DMAC

Definition at line 537 of file `vtss_security_api.h`.

7.10.2.8 vlan

`vtss_ace_vlan_t` `vtss_ace_t::vlan`

VLAN Tag

Definition at line 539 of file vtss_security_api.h.

7.10.2.9 etype

`vtss_ace_frame_etype_t` `vtss_ace_t::etype`

VTSS_ACE_TYPE_ETYPE

Definition at line 544 of file vtss_security_api.h.

7.10.2.10 llc

`vtss_ace_frame_llc_t` `vtss_ace_t::llc`

VTSS_ACE_TYPE_LLCC

Definition at line 545 of file vtss_security_api.h.

7.10.2.11 snap

`vtss_ace_frame_snap_t` `vtss_ace_t::snap`

VTSS_ACE_TYPE_SNAP

Definition at line 546 of file vtss_security_api.h.

7.10.2.12 arp

`vtss_ace_frame_arp_t` `vtss_ace_t::arp`

VTSS_ACE_TYPE_ARP

Definition at line 547 of file vtss_security_api.h.

7.10.2.13 ipv4

```
vtss_ace_frame_ipv4_t vtss_ace_t::ipv4
```

VTSS_ACE_TYPE_IPV4

Definition at line 548 of file vtss_security_api.h.

7.10.2.14 ipv6

```
vtss_ace_frame_ipv6_t vtss_ace_t::ipv6
```

VTSS_ACE_TYPE_IPV6

Definition at line 549 of file vtss_security_api.h.

7.10.2.15 frame

```
union { ... } vtss_ace_t::frame
```

Frame type specific data

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

- vtss_api/include/[vtss_security_api.h](#)

7.11 vtss_ace_vlan_t Struct Reference

ACE VLAN information.

```
#include <vtss_security_api.h>
```

Data Fields

- [vtss_ace_vid_t vid](#)
- [vtss_ace_u8_t usr_prio](#)
- [vtss_ace_bit_t cfi](#)

7.11.1 Detailed Description

ACE VLAN information.

Definition at line 411 of file vtss_security_api.h.

7.11.2 Field Documentation

7.11.2.1 vid

`vtss_ace_vid_t` `vtss_ace_vlan_t::vid`

VLAN ID (12 bit)

Definition at line 413 of file `vtss_security_api.h`.

7.11.2.2 usr_prio

`vtss_ace_u8_t` `vtss_ace_vlan_t::usr_prio`

User priority/PCP (3 bit)

Definition at line 414 of file `vtss_security_api.h`.

7.11.2.3 cfi

`vtss_ace_bit_t` `vtss_ace_vlan_t::cfi`

CFI/DEI

Definition at line 415 of file `vtss_security_api.h`.

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

- `vtss_api/include/vtss_security_api.h`

7.12 `vtss_acl_action_t` Struct Reference

ACL Action.

```
#include <vtss_security_api.h>
```

Data Fields

- `BOOL cpu`
- `BOOL cpu_once`
- `vtss_packet_rx_queue_t cpu_queue`
- `BOOL police`
- `vtss_acl_policer_no_t policer_no`
- `BOOL learn`
- `BOOL forward`
- `BOOL port_forward`
- `vtss_port_no_t port_no`
- `BOOL irq_trigger`

7.12.1 Detailed Description

ACL Action.

Definition at line 238 of file vtss_security_api.h.

7.12.2 Field Documentation

7.12.2.1 `cpu`

`BOOL vtss_acl_action_t::cpu`

Forward to CPU

Definition at line 240 of file vtss_security_api.h.

7.12.2.2 `cpu_once`

`BOOL vtss_acl_action_t::cpu_once`

Only first frame forwarded to CPU

Definition at line 241 of file vtss_security_api.h.

7.12.2.3 `cpu_queue`

`vtss_packet_rx_queue_t vtss_acl_action_t::cpu_queue`

CPU queue

Definition at line 242 of file vtss_security_api.h.

7.12.2.4 police

`BOOL vtss_acl_action_t::police`

Enable policer

Definition at line 243 of file vtss_security_api.h.

7.12.2.5 policer_no

`vtss_acl_policer_no_t vtss_acl_action_t::policer_no`

Policer number

Definition at line 244 of file vtss_security_api.h.

7.12.2.6 learn

`BOOL vtss_acl_action_t::learn`

Allow learning

Definition at line 251 of file vtss_security_api.h.

7.12.2.7 forward

`BOOL vtss_acl_action_t::forward`

Allow forwarding

Definition at line 253 of file vtss_security_api.h.

7.12.2.8 port_forward

`BOOL vtss_acl_action_t::port_forward`

Forward to specific port

Definition at line 254 of file vtss_security_api.h.

7.12.2.9 port_no

`vtss_port_no_t` `vtss_acl_action_t::port_no`

Specified port

Definition at line 255 of file `vtss_security_api.h`.

7.12.2.10 irq_trigger

`BOOL` `vtss_acl_action_t::irq_trigger`

Trigger interrupt against CPU.

Definition at line 267 of file `vtss_security_api.h`.

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

- `vtss_api/include/vtss_security_api.h`

7.13 vtss_acl_policer_conf_t Struct Reference

ACL policer configuration.

```
#include <vtss_security_api.h>
```

Data Fields

- `vtss_packet_rate_t rate`

7.13.1 Detailed Description

ACL policer configuration.

Definition at line 155 of file `vtss_security_api.h`.

7.13.2 Field Documentation

7.13.2.1 rate

`vtss_packet_rate_t vtss_acl_policer_conf_t::rate`

Packet rate

Definition at line 160 of file `vtss_security_api.h`.

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

- `vtss_api/include/vtss_security_api.h`

7.14 `vtss_acl_port_conf_t` Struct Reference

ACL port configuration.

```
#include <vtss_security_api.h>
```

Data Fields

- `vtss_acl_policy_no_t policy_no`
- `vtss_acl_action_t action`

7.14.1 Detailed Description

ACL port configuration.

Definition at line 276 of file `vtss_security_api.h`.

7.14.2 Field Documentation

7.14.2.1 `policy_no`

`vtss_acl_policy_no_t vtss_acl_port_conf_t::policy_no`

Policy number

Definition at line 278 of file `vtss_security_api.h`.

7.14.2.2 action

```
vtss_acl_action_t vtss_acl_port_conf_t::action
```

Action

Definition at line 279 of file vtss_security_api.h.

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

- vtss_api/include/[vtss_security_api.h](#)

7.15 vtss_aggr_mode_t Struct Reference

Aggregation traffic distribution mode.

```
#include <l2_types.h>
```

Data Fields

- `BOOL smac_enable`
- `BOOL dmac_enable`
- `BOOL sip_dip_enable`
- `BOOL sport_dport_enable`

7.15.1 Detailed Description

Aggregation traffic distribution mode.

Definition at line 39 of file l2_types.h.

7.15.2 Field Documentation

7.15.2.1 smac_enable

```
BOOL vtss_aggr_mode_t::smac_enable
```

Source MAC address

Definition at line 41 of file l2_types.h.

7.15.2.2 dmac_enable

`BOOL vtss_aggr_mode_t::dmac_enable`

Destination MAC address

Definition at line 42 of file l2_types.h.

7.15.2.3 sip_dip_enable

`BOOL vtss_aggr_mode_t::sip_dip_enable`

Source and destination IP address

Definition at line 43 of file l2_types.h.

7.15.2.4 sport_dport_enable

`BOOL vtss_aggr_mode_t::sport_dport_enable`

Source and destination UDP/TCP port

Definition at line 44 of file l2_types.h.

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

- [vtss_api/include/vtss/api/l2_types.h](#)

7.16 vtss_aneg_t Struct Reference

Auto negotiation struct.

```
#include <types.h>
```

Data Fields

- [BOOL obey_pause](#)
- [BOOL generate_pause](#)

7.16.1 Detailed Description

Auto negotiation struct.

Definition at line 482 of file types.h.

7.16.2 Field Documentation

7.16.2.1 obey_pause

`BOOL vtss_aneg_t::obey_pause`

This port should obey PAUSE frames

Definition at line 484 of file types.h.

7.16.2.2 generate_pause

`BOOL vtss_aneg_t::generate_pause`

Link partner obeys PAUSE frames

Definition at line 485 of file types.h.

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

- `vtss_api/include/vtss/api/types.h`

7.17 vtss_api_lock_t Struct Reference

API lock structure.

```
#include <vtss_misc_api.h>
```

Data Fields

- `vtss_inst_t inst`
- `const char * function`
- `const char * file`
- `int line`

7.17.1 Detailed Description

API lock structure.

Definition at line 285 of file vtss_misc_api.h.

7.17.2 Field Documentation

7.17.2.1 inst

`vtss_inst_t vtss_api_lock_t::inst`

Target instance reference

Definition at line 286 of file `vtss_misc_api.h`.

7.17.2.2 function

`const char* vtss_api_lock_t::function`

Function name

Definition at line 287 of file `vtss_misc_api.h`.

7.17.2.3 file

`const char* vtss_api_lock_t::file`

File name

Definition at line 288 of file `vtss_misc_api.h`.

7.17.2.4 line

`int vtss_api_lock_t::line`

Line number

Definition at line 289 of file `vtss_misc_api.h`.

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

- `vtss_api/include/vtss_misc_api.h`

7.18 vtss_basic_counters_t Struct Reference

Basic counters structure.

```
#include <vtss_port_api.h>
```

Data Fields

- [u32 rx_frames](#)
- [u32 tx_frames](#)

7.18.1 Detailed Description

Basic counters structure.

Definition at line 377 of file vtss_port_api.h.

7.18.2 Field Documentation

7.18.2.1 rx_frames

```
u32 vtss_basic_counters_t::rx_frames
```

Rx frames

Definition at line 379 of file vtss_port_api.h.

7.18.2.2 tx_frames

```
u32 vtss_basic_counters_t::tx_frames
```

Tx frames

Definition at line 380 of file vtss_port_api.h.

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

- vtss_api/include/[vtss_port_api.h](#)

7.19 vtss_chip_id_t Struct Reference

Chip ID.

```
#include <vtss_misc_api.h>
```

Data Fields

- [u16 part_number](#)
- [u16 revision](#)

7.19.1 Detailed Description

Chip ID.

Definition at line 401 of file [vtss_misc_api.h](#).

7.19.2 Field Documentation

7.19.2.1 part_number

[u16 vtss_chip_id_t::part_number](#)

BCD encoded part number

Definition at line 403 of file [vtss_misc_api.h](#).

7.19.2.2 revision

[u16 vtss_chip_id_t::revision](#)

Chip revision

Definition at line 404 of file [vtss_misc_api.h](#).

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

- [vtss_api/include/vtss_misc_api.h](#)

7.20 vtss_counter_pair_t Struct Reference

Counter pair.

```
#include <types.h>
```

Data Fields

- [vtss_counter_t frames](#)
- [vtss_counter_t bytes](#)

7.20.1 Detailed Description

Counter pair.

Definition at line 1111 of file types.h.

7.20.2 Field Documentation

7.20.2.1 frames

`vtss_counter_t vtss_counter_pair_t::frames`

Number of frames

Definition at line 1112 of file types.h.

7.20.2.2 bytes

`vtss_counter_t vtss_counter_pair_t::bytes`

Number of bytes

Definition at line 1113 of file types.h.

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

- `vtss_api/include/vtss/api/types.h`

7.21 vtss_debug_info_t Struct Reference

Debug information structure.

```
#include <vtss_misc_api.h>
```

Data Fields

- `vtss_debug_layer_t layer`
- `vtss_debug_group_t group`
- `vtss_chip_no_t chip_no`
- `BOOL port_list[VTSS_PORT_ARRAY_SIZE]`
- `BOOL full`
- `BOOL clear`
- `BOOL vml_format`

7.21.1 Detailed Description

Debug information structure.

Definition at line 243 of file vtss_misc_api.h.

7.21.2 Field Documentation

7.21.2.1 layer

```
vtss_debug_layer_t vtss_debug_info_t::layer
```

Layer

Definition at line 244 of file vtss_misc_api.h.

7.21.2.2 group

```
vtss_debug_group_t vtss_debug_info_t::group
```

Function group

Definition at line 245 of file vtss_misc_api.h.

7.21.2.3 chip_no

```
vtss_chip_no_t vtss_debug_info_t::chip_no
```

Chip number, multi-chip targets

Definition at line 246 of file vtss_misc_api.h.

7.21.2.4 port_list

```
BOOL vtss_debug_info_t::port_list[VTSS_PORT_ARRAY_SIZE]
```

Port list

Definition at line 247 of file vtss_misc_api.h.

7.21.2.5 full

`BOOL vtss_debug_info_t::full`

Full information dump

Definition at line 248 of file vtss_misc_api.h.

7.21.2.6 clear

`BOOL vtss_debug_info_t::clear`

Clear counters

Definition at line 249 of file vtss_misc_api.h.

7.21.2.7 vml_format

`BOOL vtss_debug_info_t::vml_format`

VML format register dump

Definition at line 250 of file vtss_misc_api.h.

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

- vtss_api/include/[vtss_misc_api.h](#)

7.22 vtss_debug_lock_t Struct Reference

API debug lock structure.

```
#include <vtss_misc_api.h>
```

Data Fields

- [vtss_chip_no_t chip_no](#)

7.22.1 Detailed Description

API debug lock structure.

Definition at line 307 of file vtss_misc_api.h.

7.22.2 Field Documentation

7.22.2.1 chip_no

`vtss_chip_no_t vtss_debug_lock_t::chip_no`

Chip number (if multi-chip instance).

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

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

- `vtss_api/include/vtss_misc_api.h`

7.23 `vtss_dgroup_port_conf_t` Struct Reference

Destination group port configuration.

```
#include <vtss_l2_api.h>
```

Data Fields

- `vtss_dgroup_no_t dgroup_no`

7.23.1 Detailed Description

Destination group port configuration.

Definition at line 1394 of file `vtss_l2_api.h`.

7.23.2 Field Documentation

7.23.2.1 dgroup_no

`vtss_dgroup_no_t vtss_dgroup_port_conf_t::dgroup_no`

Destination port group

Definition at line 1395 of file `vtss_l2_api.h`.

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

- `vtss_api/include/vtss_l2_api.h`

7.24 vtss_dlb_policer_conf_t Struct Reference

Dual leaky buckets policer configuration.

```
#include <vtss_qos_api.h>
```

Data Fields

- `vtss_policer_type_t type`
- `BOOL enable`
- `BOOL cm`
- `BOOL cf`
- `BOOL line_rate`
- `vtss_bitrate_t cir`
- `vtss_burst_level_t cbs`
- `vtss_bitrate_t eir`
- `vtss_burst_level_t ebs`

7.24.1 Detailed Description

Dual leaky buckets policer configuration.

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

7.24.2 Field Documentation

7.24.2.1 type

```
vtss_policer_type_t vtss_dlb_policer_conf_t::type
```

Policer type

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

7.24.2.2 enable

```
BOOL vtss_dlb_policer_conf_t::enable
```

Enable/disable policer

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

7.24.2.3 cm

`BOOL vtss_dlb_policer_conf_t::cm`

Colour Mode (TRUE means colour aware)

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

7.24.2.4 cf

`BOOL vtss_dlb_policer_conf_t::cf`

Coupling Flag

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

7.24.2.5 line_rate

`BOOL vtss_dlb_policer_conf_t::line_rate`

Line rate policing (default is data rate policing)

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

7.24.2.6 cir

`vtss_bitrate_t vtss_dlb_policer_conf_t::cir`

Committed Information Rate

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

7.24.2.7 cbs

`vtss_burst_level_t vtss_dlb_policer_conf_t::cbs`

Committed Burst Size

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

7.24.2.8 eir

`vtss_bitrate_t vtss_dlb_policer_conf_t::eir`

Excess Information Rate

Definition at line 247 of file vtss_qos_api.h.

7.24.2.9 ebs

`vtss_burst_level_t vtss_dlb_policer_conf_t::ebs`

Excess Burst Size

Definition at line 248 of file vtss_qos_api.h.

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

- vtss_api/include/[vtss_qos_api.h](#)

7.25 vtss_ece_action_t Struct Reference

ECE action.

```
#include <vtss_evc_api.h>
```

Data Fields

- [vtss_ece_dir_t](#) dir
- [vtss_ece_pop_tag_t](#) pop_tag
- [vtss_ece_outer_tag_t](#) outer_tag
- [vtss_ece_inner_tag_t](#) inner_tag
- [vtss_evc_policer_id_t](#) policer_id
- [vtss_evc_id_t](#) evc_id
- [vtss_acl_policy_no_t](#) policy_no

7.25.1 Detailed Description

ECE action.

Definition at line 557 of file vtss_evc_api.h.

7.25.2 Field Documentation

7.25.2.1 dir

`vtss_ece_dir_t` `vtss_ece_action_t::dir`

Traffic direction

Definition at line 558 of file vtss_evc_api.h.

7.25.2.2 pop_tag

`vtss_ece_pop_tag_t` `vtss_ece_action_t::pop_tag`

Ingress VLAN popping

Definition at line 563 of file vtss_evc_api.h.

7.25.2.3 outer_tag

`vtss_ece_outer_tag_t` `vtss_ece_action_t::outer_tag`

Egress outer VLAN tag (always present)

Definition at line 564 of file vtss_evc_api.h.

7.25.2.4 inner_tag

`vtss_ece_inner_tag_t` `vtss_ece_action_t::inner_tag`

Egress inner VLAN tag (optional)

Definition at line 566 of file vtss_evc_api.h.

7.25.2.5 policer_id

`vtss_evc_policer_id_t` `vtss_ece_action_t::policer_id`

Policer ID

Definition at line 567 of file vtss_evc_api.h.

7.25.2.6 evc_id

`vtss_evc_id_t` `vtss_ece_action_t::evc_id`

EVC ID

Definition at line 569 of file `vtss_evc_api.h`.

7.25.2.7 policy_no

`vtss_acl_policy_no_t` `vtss_ece_action_t::policy_no`

ACL policy number

Definition at line 570 of file `vtss_evc_api.h`.

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

- `vtss_api/include/vtss_evc_api.h`

7.26 vtss_ece_frame_ipv4_t Struct Reference

ECE IPv4 information.

```
#include <vtss_evc_api.h>
```

Data Fields

- `vtss_vcap_vr_t dscp`

7.26.1 Detailed Description

ECE IPv4 information.

Definition at line 461 of file `vtss_evc_api.h`.

7.26.2 Field Documentation

7.26.2.1 dscp

`vtss_vcap_vr_t vtss_ece_frame_ipv4_t::dscp`

DSCP field (6 bit)

Definition at line 462 of file `vtss_evc_api.h`.

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

- `vtss_api/include/vtss_evc_api.h`

7.27 `vtss_ece_frame_ipv6_t` Struct Reference

ECE IPv6 information.

```
#include <vtss_evc_api.h>
```

Data Fields

- `vtss_vcap_vr_t dscp`

7.27.1 Detailed Description

ECE IPv6 information.

Definition at line 476 of file `vtss_evc_api.h`.

7.27.2 Field Documentation

7.27.2.1 dscp

`vtss_vcap_vr_t vtss_ece_frame_ipv6_t::dscp`

DSCP field (6 bit)

Definition at line 477 of file `vtss_evc_api.h`.

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

- `vtss_api/include/vtss_evc_api.h`

7.28 vtss_ece_inner_tag_t Struct Reference

ECE inner tag.

```
#include <vtss_evc_api.h>
```

Data Fields

- `vtss_ece_inner_tag_type_t type`
- `vtss_vid_t vid`
- `BOOL pcp_dei_preserve`
- `vtss_tagprio_t pcp`
- `vtss_dei_t dei`

7.28.1 Detailed Description

ECE inner tag.

Definition at line 538 of file vtss_evc_api.h.

7.28.2 Field Documentation

7.28.2.1 type

```
vtss_ece_inner_tag_type_t vtss_ece_inner_tag_t::type
```

Tag type

Definition at line 540 of file vtss_evc_api.h.

7.28.2.2 vid

```
vtss_vid_t vtss_ece_inner_tag_t::vid
```

VLAN ID

Definition at line 541 of file vtss_evc_api.h.

7.28.2.3 pcp_dei_preserve

`BOOL vtss_ece_inner_tag_t::pcp_dei_preserve`

Preserved or explicit PCP/DEI values

Definition at line 546 of file `vtss_evc_api.h`.

7.28.2.4 pcp

`vtss_tagprio_t vtss_ece_inner_tag_t::pcp`

PCP value

Definition at line 548 of file `vtss_evc_api.h`.

7.28.2.5 dei

`vtss_dei_t vtss_ece_inner_tag_t::dei`

DEI value

Definition at line 552 of file `vtss_evc_api.h`.

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

- `vtss_api/include/vtss_evc_api.h`

7.29 vtss_ece_key_t Struct Reference

ECE key.

```
#include <vtss_evc_api.h>
```

Data Fields

- `vtss_ece_port_t port_list [VTSS_PORT_ARRAY_SIZE]`
- `vtss_ece_mac_t mac`
- `vtss_ece_tag_t tag`
- `vtss_ece_tag_t inner_tag`
- `vtss_ece_type_t type`
- union {
 - `vtss_ece_frame_ipv4_t ipv4`
 - `vtss_ece_frame_ipv6_t ipv6`}

7.29.1 Detailed Description

ECE key.

Definition at line 490 of file vtss_evc_api.h.

7.29.2 Field Documentation

7.29.2.1 port_list

```
vtss_ece_port_t vtss_ece_key_t::port_list[VTSS_PORT_ARRAY_SIZE]
```

UNI port list

Definition at line 492 of file vtss_evc_api.h.

7.29.2.2 mac

```
vtss_ece_mac_t vtss_ece_key_t::mac
```

MAC header

Definition at line 493 of file vtss_evc_api.h.

7.29.2.3 tag

```
vtss_ece_tag_t vtss_ece_key_t::tag
```

Tag

Definition at line 494 of file vtss_evc_api.h.

7.29.2.4 inner_tag

```
vtss_ece_tag_t vtss_ece_key_t::inner_tag
```

Inner tag

Definition at line 496 of file vtss_evc_api.h.

7.29.2.5 type

`vtss_ece_type_t vtss_ece_key_t::type`

Frame type

Definition at line 498 of file `vtss_evc_api.h`.

7.29.2.6 ipv4

`vtss_ece_frame_ipv4_t vtss_ece_key_t::ipv4`

`VTSS_ECE_TYPE_IPV4`

Definition at line 511 of file `vtss_evc_api.h`.

7.29.2.7 ipv6

`vtss_ece_frame_ipv6_t vtss_ece_key_t::ipv6`

`VTSS_ECE_TYPE_IPV6`

Definition at line 512 of file `vtss_evc_api.h`.

7.29.2.8 frame

`union { ... } vtss_ece_key_t::frame`

Frame type specific data

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

- `vtss_api/include/vtss_evc_api.h`

7.30 `vtss_ece_mac_t` Struct Reference

ECE MAC information.

```
#include <vtss_evc_api.h>
```

Data Fields

- [vtss_vcap_u48_t dmac](#)

7.30.1 Detailed Description

ECE MAC information.

Definition at line 420 of file vtss_evc_api.h.

7.30.2 Field Documentation

7.30.2.1 dmac

[vtss_vcap_u48_t](#) vtss_ece_mac_t::dmac

DMAC

Definition at line 423 of file vtss_evc_api.h.

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

- [vtss_api/include/vtss_evc_api.h](#)

7.31 vtss_ece_outer_tag_t Struct Reference

ECE outer tag.

```
#include <vtss_evc_api.h>
```

Data Fields

- [BOOL enable](#)
- [vtss_vid_t vid](#)
- [BOOL pcp_dei_preserve](#)
- [vtss_tagprio_t pcp](#)
- [vtss_dei_t dei](#)

7.31.1 Detailed Description

ECE outer tag.

Definition at line 517 of file vtss_evc_api.h.

7.31.2 Field Documentation

7.31.2.1 enable

`BOOL vtss_ece_outer_tag_t::enable`

VLAN tag (VTSS_ECE_DIR_NNI_TO_UNI only)

Definition at line 519 of file vtss_evc_api.h.

7.31.2.2 vid

`vtss_vid_t vtss_ece_outer_tag_t::vid`

VLAN ID (VTSS_ECE_DIR_NNI_TO_UNI only)

Definition at line 521 of file vtss_evc_api.h.

7.31.2.3 pcp_dei_preserve

`BOOL vtss_ece_outer_tag_t::pcp_dei_preserve`

Preserved or explicit PCP/DEI values

Definition at line 527 of file vtss_evc_api.h.

7.31.2.4 pcp

`vtss_tagprio_t vtss_ece_outer_tag_t::pcp`

PCP value

Definition at line 529 of file vtss_evc_api.h.

7.31.2.5 dei

`vtss_dei_t vtss_ece_outer_tag_t::dei`

DEI value (ignored if colouring enabled)

Definition at line 533 of file vtss_evc_api.h.

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

- `vtss_api/include/vtss_evc_api.h`

7.32 vtss_ece_t Struct Reference

EVC Control Entry.

```
#include <vtss_evc_api.h>
```

Data Fields

- `vtss_ece_id_t id`
- `vtss_ece_key_t key`
- `vtss_ece_action_t action`

7.32.1 Detailed Description

EVC Control Entry.

Definition at line 582 of file vtss_evc_api.h.

7.32.2 Field Documentation

7.32.2.1 id

`vtss_ece_id_t vtss_ece_t::id`

Entry ID

Definition at line 583 of file vtss_evc_api.h.

7.32.2.2 key

`vtss_ece_key_t` `vtss_ece_t::key`

ECE key

Definition at line 584 of file vtss_evc_api.h.

7.32.2.3 action

`vtss_ece_action_t` `vtss_ece_t::action`

ECE action

Definition at line 585 of file vtss_evc_api.h.

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

- vtss_api/include/[vtss_evc_api.h](#)

7.33 vtss_ece_tag_t Struct Reference

ECE tag information.

```
#include <vtss_evc_api.h>
```

Data Fields

- `vtss_vcap_vr_t vid`
- `vtss_vcap_u8_t pcp`
- `vtss_vcap_bit_t dei`
- `vtss_vcap_bit_t tagged`
- `vtss_vcap_bit_t s_tagged`

7.33.1 Detailed Description

ECE tag information.

Definition at line 433 of file vtss_evc_api.h.

7.33.2 Field Documentation

7.33.2.1 vid

`vtss_vcap_vr_t` `vtss_ece_tag_t::vid`

VLAN ID (12 bit)

Definition at line 435 of file vtss_evc_api.h.

7.33.2.2 pcp

`vtss_vcap_u8_t` `vtss_ece_tag_t::pcp`

PCP (3 bit)

Definition at line 436 of file vtss_evc_api.h.

7.33.2.3 dei

`vtss_vcap_bit_t` `vtss_ece_tag_t::dei`

DEI

Definition at line 437 of file vtss_evc_api.h.

7.33.2.4 tagged

`vtss_vcap_bit_t` `vtss_ece_tag_t::tagged`

Tagged/untagged frame

Definition at line 438 of file vtss_evc_api.h.

7.33.2.5 s_tagged

`vtss_vcap_bit_t` `vtss_ece_tag_t::s_tagged`

S-tagged/C-tagged frame

Definition at line 439 of file vtss_evc_api.h.

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

- vtss_api/include/[vtss_evc_api.h](#)

7.34 vtss_eee_port_conf_t Struct Reference

EEE port configuration.

```
#include <vtss_misc_api.h>
```

Data Fields

- `BOOL eee_ena`
- `u8 eee_fast_queues`
- `u16 tx_tw`
- `u8 lp_advertisement`
- `BOOL optimized_for_power`

7.34.1 Detailed Description

EEE port configuration.

Definition at line 1219 of file `vtss_misc_api.h`.

7.34.2 Field Documentation

7.34.2.1 eee_ena

```
BOOL vtss_eee_port_conf_t::eee_ena
```

Enable EEE

Definition at line 1221 of file `vtss_misc_api.h`.

7.34.2.2 eee_fast_queues

```
u8 vtss_eee_port_conf_t::eee_fast_queues
```

Queues set in this mask will activate egress path as soon as any data is available. Vector for enabling fast queues.
bit 0 = queue 0, bit 1 = queue 1 and so on.

Definition at line 1222 of file `vtss_misc_api.h`.

7.34.2.3 tx_tw

```
u16 vtss_eee_port_conf_t::tx_tw
```

Time from path is activated until frame transmission restarted.

Definition at line 1223 of file vtss_misc_api.h.

7.34.2.4 lp_advertisement

```
u8 vtss_eee_port_conf_t::lp_advertisement
```

Link partner EEE advertisement. Bit 0 = Link partner advertises 100BASE-T capability. Bit 1 = Link partner advertises 1000BASE-T capability.

Definition at line 1224 of file vtss_misc_api.h.

7.34.2.5 optimized_for_power

```
BOOL vtss_eee_port_conf_t::optimized_for_power
```

EEE can be optimized for either most power savings or least traffic latency

Definition at line 1226 of file vtss_misc_api.h.

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

- vtss_api/include/[vtss_misc_api.h](#)

7.35 vtss_eee_port_counter_t Struct Reference

EEE port counters (JR only)

```
#include <vtss_misc_api.h>
```

Data Fields

- [BOOL fill_level_get](#)
- [u32 fill_level_thres](#)
- [u32 fill_level](#)
- [BOOL tx_out_bytes_get](#)
- [u32 tx_out_bytes](#)

7.35.1 Detailed Description

EEE port counters (JR only)

Definition at line 1246 of file vtss_misc_api.h.

7.35.2 Field Documentation

7.35.2.1 fill_level_get

```
BOOL vtss_eee_port_counter_t::fill_level_get
```

[IN] FALSE => Don't get fill level. TRUE => Get fill level.

Definition at line 1248 of file vtss_misc_api.h.

7.35.2.2 fill_level_thres

```
u32 vtss_eee_port_counter_t::fill_level_thres
```

[IN] Stop iterating over queues when fill level exceeds this value.

Definition at line 1249 of file vtss_misc_api.h.

7.35.2.3 fill_level

```
u32 vtss_eee_port_counter_t::fill_level
```

[OUT] Accumulated fill level, updated by API if [fill_level_get](#) is TRUE.

Definition at line 1250 of file vtss_misc_api.h.

7.35.2.4 tx_out_bytes_get

```
BOOL vtss_eee_port_counter_t::tx_out_bytes_get
```

[IN] FALSE => Don't get transmitted bytes. TRUE => Get tx'd bytes.

Definition at line 1251 of file vtss_misc_api.h.

7.35.2.5 tx_out_bytes

```
u32 vtss_eee_port_counter_t::tx_out_bytes
```

[OUT] Transmitted number of bytes, updated by API if [tx_out_bytes_get](#) is TRUE.

Definition at line 1252 of file [vtss_misc_api.h](#).

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

- [vtss_api/include/vtss_misc_api.h](#)

7.36 vtss_eee_port_state_t Struct Reference

EEE port state (JR only)

```
#include <vtss_misc_api.h>
```

Data Fields

- [vtss_eee_state_select_t select](#)
- [u32 val](#)

7.36.1 Detailed Description

EEE port state (JR only)

Definition at line 1239 of file [vtss_misc_api.h](#).

7.36.2 Field Documentation

7.36.2.1 select

```
vtss\_eee\_state\_select\_t vtss_eee_port_state_t::select
```

State to change.

Definition at line 1241 of file [vtss_misc_api.h](#).

7.36.2.2 val

```
u32 vtss_eee_port_state_t::val
```

New value to apply. Interpretation depends on [select](#).

Definition at line 1242 of file [vtss_misc_api.h](#).

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

- [vtss_api/include/vtss_misc_api.h](#)

7.37 vtss_eps_port_conf_t Struct Reference

Port protection configuration.

```
#include <vtss_l2_api.h>
```

Data Fields

- [vtss_eps_port_type_t type](#)
- [vtss_port_no_t port_no](#)

7.37.1 Detailed Description

Port protection configuration.

Definition at line 2141 of file [vtss_l2_api.h](#).

7.37.2 Field Documentation

7.37.2.1 type

```
vtss_eps_port_type_t vtss_eps_port_conf_t::type
```

Protection type

Definition at line 2143 of file [vtss_l2_api.h](#).

7.37.2.2 port_no

`vtss_port_no_t vtss_eps_port_conf_t::port_no`

Protection port or VTSS_PORT_NO_NONE

Definition at line 2144 of file vtss_l2_api.h.

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

- `vtss_api/include/vtss_l2_api.h`

7.38 vtss_evc_conf_t Struct Reference

EVC configuration (excluding UNIs)

```
#include <vtss_evc_api.h>
```

Data Fields

- `vtss_evc_policer_id_t policer_id`
- `BOOL learning`
- `struct {
 vtss_evc_pb_conf_t pb
} network`

7.38.1 Detailed Description

EVC configuration (excluding UNIs)

Definition at line 309 of file vtss_evc_api.h.

7.38.2 Field Documentation

7.38.2.1 policer_id

`vtss_evc_policer_id_t vtss_evc_conf_t::policer_id`

Policer ID

Definition at line 311 of file vtss_evc_api.h.

7.38.2.2 learning

`BOOL vtss_evc_conf_t::learning`

Enable/disable learning

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

7.38.2.3 pb

`vtss_evc_pb_conf_t vtss_evc_conf_t::pb`

PB specific configuration

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

7.38.2.4 network

`struct { ... } vtss_evc_conf_t::network`

Network specific configuration

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

- `vtss_api/include/vtss_evc_api.h`

7.39 vtss_evc_counters_t Struct Reference

EVC/ECE counters.

```
#include <types.h>
```

Data Fields

- `vtss_counter_pair_t rx_green`
- `vtss_counter_pair_t rx_yellow`
- `vtss_counter_pair_t rx_red`
- `vtss_counter_pair_t rx_discard`
- `vtss_counter_pair_t tx_discard`
- `vtss_counter_pair_t tx_green`
- `vtss_counter_pair_t tx_yellow`

7.39.1 Detailed Description

EVC/ECE counters.

Definition at line 1127 of file types.h.

7.39.2 Field Documentation

7.39.2.1 rx_green

```
vtss_counter_pair_t vtss_evc_counters_t::rx_green
```

Rx green frames/bytes

Definition at line 1128 of file types.h.

7.39.2.2 rx_yellow

```
vtss_counter_pair_t vtss_evc_counters_t::rx_yellow
```

Rx yellow frames/bytes

Definition at line 1129 of file types.h.

7.39.2.3 rx_red

```
vtss_counter_pair_t vtss_evc_counters_t::rx_red
```

Rx red frames/bytes

Definition at line 1130 of file types.h.

7.39.2.4 rx_discard

```
vtss_counter_pair_t vtss_evc_counters_t::rx_discard
```

Rx discarded frames/bytes

Definition at line 1131 of file types.h.

7.39.2.5 tx_discard

```
vtss_counter_pair_t vtss_evc_counters_t::tx_discard
```

Tx discarded frames/bytes

Definition at line 1132 of file types.h.

7.39.2.6 tx_green

```
vtss_counter_pair_t vtss_evc_counters_t::tx_green
```

Tx green frames/bytes

Definition at line 1133 of file types.h.

7.39.2.7 tx_yellow

```
vtss_counter_pair_t vtss_evc_counters_t::tx_yellow
```

Tx yellow frames/bytes

Definition at line 1134 of file types.h.

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

- vtss_api/include/vtss/api/[types.h](#)

7.40 vtss_evc_pb_conf_t Struct Reference

PB specific EVC configuration.

```
#include <vtss_evc_api.h>
```

Data Fields

- [BOOL nni \[VTSS_PORT_ARRAY_SIZE\]](#)
- [vtss_vid_t ivid](#)
- [vtss_vid_t vid](#)
- [vtss_vid_t leaf_ivid](#)
- [vtss_vid_t leaf_vid](#)
- [BOOL leaf \[VTSS_PORT_ARRAY_SIZE\]](#)

7.40.1 Detailed Description

PB specific EVC configuration.

Definition at line 275 of file vtss_evc_api.h.

7.40.2 Field Documentation

7.40.2.1 nni

```
BOOL vtss_evc_pb_conf_t::nni[VTSS_PORT_ARRAY_SIZE]
```

NNI configuration

Definition at line 276 of file vtss_evc_api.h.

7.40.2.2 ivid

```
vtss_vid_t vtss_evc_pb_conf_t::ivid
```

Internal VID

Definition at line 277 of file vtss_evc_api.h.

7.40.2.3 vid

```
vtss_vid_t vtss_evc_pb_conf_t::vid
```

NNI VID of outer tag

Definition at line 278 of file vtss_evc_api.h.

7.40.2.4 leaf_ivid

```
vtss_vid_t vtss_evc_pb_conf_t::leaf_ivid
```

Leaf internal VID

Definition at line 280 of file vtss_evc_api.h.

7.40.2.5 leaf_vid

`vtss_vid_t vtss_evc_pb_conf_t::leaf_vid`

Leaf NNI VID of outer tag

Definition at line 281 of file `vtss_evc_api.h`.

7.40.2.6 leaf

`BOOL vtss_evc_pb_conf_t::leaf[VTSS_PORT_ARRAY_SIZE]`

UNI leaf

Definition at line 282 of file `vtss_evc_api.h`.

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

- `vtss_api/include/vtss_evc_api.h`

7.41 vtss_evc_port_conf_t Struct Reference

EVC port configuration.

```
#include <vtss_evc_api.h>
```

Data Fields

- `BOOL dei_colouring`

7.41.1 Detailed Description

EVC port configuration.

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

7.41.2 Field Documentation

7.41.2.1 dei_colouring

`BOOL vtss_evc_port_conf_t::dei_colouring`

NNI: Enable colouring of DEI for received frames

Definition at line 152 of file `vtss_evc_api.h`.

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

- `vtss_api/include/vtss_evc_api.h`

7.42 vtss_ewis_aisl_cons_act_s Struct Reference

eWIS AIS-L consequent actions

```
#include <vtss_wis_api.h>
```

Data Fields

- `BOOL ais_on_los`
- `BOOL ais_on_lof`

7.42.1 Detailed Description

eWIS AIS-L consequent actions

Definition at line 77 of file `vtss_wis_api.h`.

7.42.2 Field Documentation

7.42.2.1 ais_on_los

`BOOL vtss_ewis_aisl_cons_act_s::ais_on_los`

TRUE = enable for AIS-L insertion on LOS

Definition at line 78 of file `vtss_wis_api.h`.

7.42.2.2 ais_on_lof

`BOOL vtss_ewis_aisl_cons_act_s::ais_on_lof`

TRUE = enable for AIS-L insertion on LOF

Definition at line 79 of file `vtss_wis_api.h`.

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

- `vtss_api/include/vtss_wis_api.h`

7.43 vtss_ewis_conf_s Struct Reference

eWIS configuration primitives, used to hold all the configuration parameters in the internal state in the API.

```
#include <vtss_wis_api.h>
```

Data Fields

- `BOOL ewis_init_done`
- `vtss_ewis_static_conf_t static_conf`
- `vtss_ewis_mode_t ewis_mode`
- `vtss_ewis_cons_act_t section_cons_act`
- `vtss_ewis_tti_t section_txi`
- `vtss_ewis_force_mode_t force_mode`
- `vtss_ewis_tti_t path_txi`
- `vtss_ewis_tx_oh_t tx_oh`
- `vtss_ewis_tx_oh_passthru_t tx_oh_passthru`
- `vtss_ewis_sl_conf_t exp_sl`
- `vtss_ewis_test_conf_t test_conf`
- `vtss_ewis_counter_threshold_t ewis_cntr_thresh_conf`
- `vtss_ewis_perf_mode_t perf_mode`

7.43.1 Detailed Description

eWIS configuration primitives, used to hold all the configuration parameters in the internal state in the API.

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

7.43.2 Field Documentation

7.43.2.1 ewis_init_done

`BOOL vtss_ewis_conf_s::ewis_init_done`

Indicate WIS mode is enabled

Definition at line 314 of file vtss_wis_api.h.

7.43.2.2 static_conf

`vtss_ewis_static_conf_t vtss_ewis_conf_s::static_conf`

Static configuration

Definition at line 315 of file vtss_wis_api.h.

7.43.2.3 ewis_mode

`vtss_ewis_mode_t vtss_ewis_conf_s::ewis_mode`

EWIS mode configuration

Definition at line 316 of file vtss_wis_api.h.

7.43.2.4 section_cons_act

`vtss_ewis_cons_act_t vtss_ewis_conf_s::section_cons_act`

Section consequent action configuraiton

Definition at line 317 of file vtss_wis_api.h.

7.43.2.5 section_txti

`vtss_ewis_txti_t vtss_ewis_conf_s::section_txti`

Section Trail Trace Identifier configuration

Definition at line 318 of file vtss_wis_api.h.

7.43.2.6 force_mode

`vtss_ewis_force_mode_t` `vtss_ewis_conf_s::force_mode`

Force mode configuration

Definition at line 319 of file vtss_wis_api.h.

7.43.2.7 path_txti

`vtss_ewis_tti_t` `vtss_ewis_conf_s::path_txti`

Path Trail Trace Identifier Configuration

Definition at line 320 of file vtss_wis_api.h.

7.43.2.8 tx_oh

`vtss_ewis_tx_oh_t` `vtss_ewis_conf_s::tx_oh`

Transmit Overhead

Definition at line 321 of file vtss_wis_api.h.

7.43.2.9 tx_oh_passthru

`vtss_ewis_tx_oh_passthru_t` `vtss_ewis_conf_s::tx_oh_passthru`

Transmit Overhead Passthru Configuration

Definition at line 322 of file vtss_wis_api.h.

7.43.2.10 exp_sl

`vtss_ewis_sl_conf_t` `vtss_ewis_conf_s::exp_sl`

Expected Signal Label

Definition at line 323 of file vtss_wis_api.h.

7.43.2.11 test_conf

`vtss_ewis_test_conf_t` `vtss_ewis_conf_s::test_conf`

EWIS Test Mode configuration

Definition at line 324 of file `vtss_wis_api.h`.

7.43.2.12 ewis_cntr_thresh_conf

`vtss_ewis_counter_threshold_t` `vtss_ewis_conf_s::ewis_cntr_thresh_conf`

EWIS counter threshold configuration

Definition at line 325 of file `vtss_wis_api.h`.

7.43.2.13 perf_mode

`vtss_ewis_perf_mode_t` `vtss_ewis_conf_s::perf_mode`

EWIS mode configuration

Definition at line 326 of file `vtss_wis_api.h`.

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

- `vtss_api/include/vtss_wis_api.h`

7.44 vtss_ewis_cons_act_s Struct Reference

eWIS consequent actions

```
#include <vtss_wis_api.h>
```

Data Fields

- `vtss_ewis_aisl_cons_act_t` `aisl`
- `vtss_ewis_rdil_cons_act_t` `rdil`
- `vtss_ewis_fault_cons_act_t` `fault`

7.44.1 Detailed Description

eWIS consequent actions

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

7.44.2 Field Documentation

7.44.2.1 aisl

`vtss_ewis_aisl_cons_act_t` `vtss_ewis_cons_act_s::aisl`

AIS-L consequent action configuration

Definition at line 92 of file `vtss_wis_api.h`.

7.44.2.2 rdil

`vtss_ewis_rdil_cons_act_t` `vtss_ewis_cons_act_s::rdil`

RDI-L consequent action configuration

Definition at line 93 of file `vtss_wis_api.h`.

7.44.2.3 fault

`vtss_ewis_fault_cons_act_t` `vtss_ewis_cons_act_s::fault`

FAULT condition consequent action configuration

Definition at line 94 of file `vtss_wis_api.h`.

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

- `vtss_api/include/vtss_wis_api.h`

7.45 `vtss_ewis_counter_s` Struct Reference

eWIS performance counters. These counters are free running counters that wraps to zero.

```
#include <vtss_wis_api.h>
```

Data Fields

- `u16 pn_ebc_p`
- `u16 pf_ebc_p`
- `u32 pn_ebc_l`
- `u32 pf_ebc_l`
- `u16 pn_ebc_s`

7.45.1 Detailed Description

eWIS performance counters. These counters are free running counters that wraps to zero.

Definition at line 187 of file vtss_wis_api.h.

7.45.2 Field Documentation

7.45.2.1 pn_ebc_p

`u16 vtss_ewis_counter_s::pn_ebc_p`

Path block error count

Definition at line 188 of file vtss_wis_api.h.

7.45.2.2 pf_ebc_p

`u16 vtss_ewis_counter_s::pf_ebc_p`

Far end path block error count

Definition at line 189 of file vtss_wis_api.h.

7.45.2.3 pn_ebc_l

`u32 vtss_ewis_counter_s::pn_ebc_l`

Near end line block (BIP) error count

Definition at line 190 of file vtss_wis_api.h.

7.45.2.4 pf_ebc_l

`u32 vtss_ewis_counter_s::pf_ebc_l`

Far end line block (BIP) error count

Definition at line 191 of file vtss_wis_api.h.

7.45.2.5 pn_ebc_s

`u16 vtss_ewis_counter_s::pn_ebc_s`

Section BIP error count

Definition at line 192 of file `vtss_wis_api.h`.

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

- `vtss_api/include/vtss_wis_api.h`

7.46 vtss_ewis_counter_threshold_s Struct Reference

eWIS performance counter thresholds.

```
#include <vtss_wis_api.h>
```

Data Fields

- `u32 n_ebc_thr_s`
- `u32 n_ebc_thr_l`
- `u32 f_ebc_thr_l`
- `u32 n_ebc_thr_p`
- `u32 f_ebc_thr_p`

7.46.1 Detailed Description

eWIS performance counter thresholds.

Definition at line 269 of file `vtss_wis_api.h`.

7.46.2 Field Documentation

7.46.2.1 n_ebc_thr_s

`u32 vtss_ewis_counter_threshold_s::n_ebc_thr_s`

Section error count (B1) threshold

Definition at line 270 of file `vtss_wis_api.h`.

7.46.2.2 n_ebc_thr_l

`u32 vtss_ewis_counter_threshold_s::n_ebc_thr_l`

Near end line error count (B2) threshold

Definition at line 271 of file vtss_wis_api.h.

7.46.2.3 f_ebc_thr_l

`u32 vtss_ewis_counter_threshold_s::f_ebc_thr_l`

Far end line error count threshold

Definition at line 272 of file vtss_wis_api.h.

7.46.2.4 n_ebc_thr_p

`u32 vtss_ewis_counter_threshold_s::n_ebc_thr_p`

Path block error count (B3) threshold

Definition at line 273 of file vtss_wis_api.h.

7.46.2.5 f_ebc_thr_p

`u32 vtss_ewis_counter_threshold_s::f_ebc_thr_p`

Far end path error count threshold

Definition at line 274 of file vtss_wis_api.h.

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

- vtss_api/include/[vtss_wis_api.h](#)

7.47 vtss_ewis_defects_s Struct Reference

eWIS defects

```
#include <vtss_wis_api.h>
```

Data Fields

- `BOOL dlos_s`
- `BOOL doof_s`
- `BOOL dlaf_s`
- `BOOL dais_l`
- `BOOL drdi_l`
- `BOOL dais_p`
- `BOOL dlaf_p`
- `BOOL duneq_p`
- `BOOL drdi_p`
- `BOOL dlcd_p`
- `BOOL dpml_p`
- `BOOL dfais_p`
- `BOOL dfplm_p`
- `BOOL dfuneq_p`

7.47.1 Detailed Description

eWIS defects

Definition at line 153 of file vtss_wis_api.h.

7.47.2 Field Documentation

7.47.2.1 `dlos_s`

`BOOL vtss_ewis_defects_s::dlos_s`

Loss of signal

Definition at line 154 of file vtss_wis_api.h.

7.47.2.2 `doof_s`

`BOOL vtss_ewis_defects_s::doof_s`

Out of frame

Definition at line 155 of file vtss_wis_api.h.

7.47.2.3 dlof_s

`BOOL vtss_ewis_defects_s::dlof_s`

Loss of frame

Definition at line 156 of file vtss_wis_api.h.

7.47.2.4 dais_l

`BOOL vtss_ewis_defects_s::dais_l`

Line alarm indication signal

Definition at line 157 of file vtss_wis_api.h.

7.47.2.5 drdi_l

`BOOL vtss_ewis_defects_s::drdi_l`

Line remote defect indication

Definition at line 158 of file vtss_wis_api.h.

7.47.2.6 dais_p

`BOOL vtss_ewis_defects_s::dais_p`

Path alarm indication signal

Definition at line 159 of file vtss_wis_api.h.

7.47.2.7 dlop_p

`BOOL vtss_ewis_defects_s::dlop_p`

Loss of pointer

Definition at line 160 of file vtss_wis_api.h.

7.47.2.8 duneq_p

`BOOL vtss_ewis_defects_s::duneq_p`

Path Unequipped, not supported in 8487/8488

Definition at line 161 of file vtss_wis_api.h.

7.47.2.9 drdi_p

`BOOL vtss_ewis_defects_s::drdi_p`

Path Remote Defect Indication, not supported in 8487/8488

Definition at line 162 of file vtss_wis_api.h.

7.47.2.10 dlcd_p

`BOOL vtss_ewis_defects_s::dlcd_p`

Path loss of code-group delineation, not supported in 8492

Definition at line 163 of file vtss_wis_api.h.

7.47.2.11 dplm_p

`BOOL vtss_ewis_defects_s::dplm_p`

Path loss of code-group delineation

Definition at line 164 of file vtss_wis_api.h.

7.47.2.12 dfais_p

`BOOL vtss_ewis_defects_s::dfais_p`

far-end AIS-P/LOP-P

Definition at line 166 of file vtss_wis_api.h.

7.47.2.13 dfplm_p

`BOOL vtss_ewis_defects_s::dfplm_p`

far-end PLM-P/LCD-P defect

Definition at line 167 of file vtss_wis_api.h.

7.47.2.14 dfuneq_p

`BOOL vtss_ewis_defects_s::dfuneq_p`

Far End Path Unequipped

Definition at line 168 of file vtss_wis_api.h.

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

- vtss_api/include/[vtss_wis_api.h](#)

7.48 vtss_ewis_fault_cons_act_s Struct Reference

eWIS fault mask configuration, i.e set up which defects trigger the Fault condition

```
#include <vtss_wis_api.h>
```

Data Fields

- `BOOL fault_on_feplmp`
- `BOOL fault_on_feaisp`
- `BOOL fault_on_rdil`
- `BOOL fault_on_sef`
- `BOOL fault_on_lof`
- `BOOL fault_on_los`
- `BOOL fault_on_aisl`
- `BOOL fault_on_lcdp`
- `BOOL fault_on_plmp`
- `BOOL fault_on_aisp`
- `BOOL fault_on_lopp`

7.48.1 Detailed Description

eWIS fault mask configuration, i.e set up which defects trigger the Fault condition

Definition at line 62 of file vtss_wis_api.h.

7.48.2 Field Documentation

7.48.2.1 fault_on_feplmp

```
BOOL vtss_ewis_fault_cons_act_s::fault_on_feplmp
```

TRUE = enable fault condition on far-end PLM-P

Definition at line 63 of file vtss_wis_api.h.

7.48.2.2 fault_on_feaisp

```
BOOL vtss_ewis_fault_cons_act_s::fault_on_feaisp
```

TRUE = enable fault condition on far-end AIS-P

Definition at line 64 of file vtss_wis_api.h.

7.48.2.3 fault_on_rdil

```
BOOL vtss_ewis_fault_cons_act_s::fault_on_rdil
```

TRUE = enable fault condition on RDI-L

Definition at line 65 of file vtss_wis_api.h.

7.48.2.4 fault_on_sef

```
BOOL vtss_ewis_fault_cons_act_s::fault_on_sef
```

TRUE = enable fault condition on SEF

Definition at line 66 of file vtss_wis_api.h.

7.48.2.5 fault_on_lof

```
BOOL vtss_ewis_fault_cons_act_s::fault_on_lof
```

TRUE = enable fault condition on LOF

Definition at line 67 of file vtss_wis_api.h.

7.48.2.6 fault_on_los

`BOOL vtss_ewis_fault_cons_act_s::fault_on_los`

TRUE = enable fault condition on LOS

Definition at line 68 of file vtss_wis_api.h.

7.48.2.7 fault_on_aisl

`BOOL vtss_ewis_fault_cons_act_s::fault_on_aisl`

TRUE = enable fault condition on AIS-L

Definition at line 69 of file vtss_wis_api.h.

7.48.2.8 fault_on_lcdp

`BOOL vtss_ewis_fault_cons_act_s::fault_on_lcdp`

TRUE = enable fault condition on LCD-P

Definition at line 70 of file vtss_wis_api.h.

7.48.2.9 fault_on_plmp

`BOOL vtss_ewis_fault_cons_act_s::fault_on_plmp`

TRUE = enable fault condition on PLM-P

Definition at line 71 of file vtss_wis_api.h.

7.48.2.10 fault_on_aisp

`BOOL vtss_ewis_fault_cons_act_s::fault_on_aisp`

TRUE = enable fault condition on AIS-P

Definition at line 72 of file vtss_wis_api.h.

7.48.2.11 fault_on_lopp

`BOOL vtss_ewis_fault_cons_act_s::fault_on_lopp`

TRUE = enable fault condition on LOP-P

Definition at line 73 of file `vtss_wis_api.h`.

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

- `vtss_api/include/vtss_wis_api.h`

7.49 `vtss_ewis_force_mode_s` Struct Reference

eWIS force modes

```
#include <vtss_wis_api.h>
```

Data Fields

- `vtss_ewis_line_force_mode_t line_rx_force`
- `vtss_ewis_line_tx_force_mode_t line_tx_force`
- `vtss_ewis_path_force_mode_t path_force`

7.49.1 Detailed Description

eWIS force modes

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

7.49.2 Field Documentation

7.49.2.1 `line_rx_force`

`vtss_ewis_line_force_mode_t vtss_ewis_force_mode_s::line_rx_force`

Line force configuration rx direction

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

7.49.2.2 line_tx_force

`vtss_ewis_line_tx_force_mode_t vtss_ewis_force_mode_s::line_tx_force`

Line force configuration tx direction

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

7.49.2.3 path_force

`vtss_ewis_path_force_mode_t vtss_ewis_force_mode_s::path_force`

Path force configuration

Definition at line 119 of file `vtss_wis_api.h`.

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

- `vtss_api/include/vtss_wis_api.h`

7.50 vtss_ewis_line_force_mode_s Struct Reference

eWIS line force mode

```
#include <vtss_wis_api.h>
```

Data Fields

- `BOOL force_ais`
- `BOOL force_rdi`

7.50.1 Detailed Description

eWIS line force mode

Definition at line 98 of file `vtss_wis_api.h`.

7.50.2 Field Documentation

7.50.2.1 force_ais

`BOOL vtss_ewis_line_force_mode_s::force_ais`

Force AIS-L configuration

Definition at line 99 of file `vtss_wis_api.h`.

7.50.2.2 force_rdi

`BOOL vtss_ewis_line_force_mode_s::force_rdi`

Force RDI-L configuration

Definition at line 100 of file `vtss_wis_api.h`.

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

- `vtss_api/include/vtss_wis_api.h`

7.51 `vtss_ewis_line_tx_force_mode_s` Struct Reference

eWIS line TX force mode

```
#include <vtss_wis_api.h>
```

Data Fields

- `BOOL force_ais`
- `BOOL force_rdi`

7.51.1 Detailed Description

eWIS line TX force mode

Definition at line 104 of file `vtss_wis_api.h`.

7.51.2 Field Documentation

7.51.2.1 force_ais

`BOOL vtss_ewis_line_tx_force_mode_s::force_ais`

Force transmission of AIS-L in the K2 byte

Definition at line 105 of file `vtss_wis_api.h`.

7.51.2.2 force_rdi

`BOOL vtss_ewis_line_tx_force_mode_s::force_rdi`

Force transmission of RDI-L in the K2 byte

Definition at line 106 of file `vtss_wis_api.h`.

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

- `vtss_api/include/vtss_wis_api.h`

7.52 vtss_ewis_path_force_mode_s Struct Reference

eWIS path force modes

```
#include <vtss_wis_api.h>
```

Data Fields

- `BOOL force_uneq`
- `BOOL force_rdi`

7.52.1 Detailed Description

eWIS path force modes

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

7.52.2 Field Documentation

7.52.2.1 force_uneq

```
BOOL vtss_ewis_path_force_mode_s::force_uneq
```

Force UNEQ-P configuration

Definition at line 111 of file vtss_wis_api.h.

7.52.2.2 force_rdi

```
BOOL vtss_ewis_path_force_mode_s::force_rdi
```

Force RDI-P configuration

Definition at line 112 of file vtss_wis_api.h.

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

- vtss_api/include/[vtss_wis_api.h](#)

7.53 vtss_ewis_perf_mode_s Struct Reference

eWIS Mode(Bit/Block) for the Performance Monitoring Counters

```
#include <vtss_wis_api.h>
```

Data Fields

- [vtss_ewis_perf_cntr_mode_t pn_ebc_mode_s](#)
- [vtss_ewis_perf_cntr_mode_t pn_ebc_mode_l](#)
- [vtss_ewis_perf_cntr_mode_t pf_ebc_mode_l](#)
- [vtss_ewis_perf_cntr_mode_t pn_ebc_mode_p](#)
- [vtss_ewis_perf_cntr_mode_t pf_ebc_mode_p](#)

7.53.1 Detailed Description

eWIS Mode(Bit/Block) for the Performance Monitoring Counters

Definition at line 129 of file vtss_wis_api.h.

7.53.2 Field Documentation

7.53.2.1 pn_ebc_mode_s

`vtss_ewis_perf_cntr_mode_t` `vtss_ewis_perf_mode_s::pn_ebc_mode_s`

Near end block (B1)

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

7.53.2.2 pn_ebc_mode_l

`vtss_ewis_perf_cntr_mode_t` `vtss_ewis_perf_mode_s::pn_ebc_mode_l`

Near end line block (BIP) error count (B2)

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

7.53.2.3 pf_ebc_mode_l

`vtss_ewis_perf_cntr_mode_t` `vtss_ewis_perf_mode_s::pf_ebc_mode_l`

Far end line block (BIP) error count (REI-L)

Definition at line 132 of file `vtss_wis_api.h`.

7.53.2.4 pn_ebc_mode_p

`vtss_ewis_perf_cntr_mode_t` `vtss_ewis_perf_mode_s::pn_ebc_mode_p`

Path block error count (B3)

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

7.53.2.5 pf_ebc_mode_p

`vtss_ewis_perf_cntr_mode_t` `vtss_ewis_perf_mode_s::pf_ebc_mode_p`

Far end path block error count (REI-P)

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

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

- `vtss_api/include/vtss_wis_api.h`

7.54 vtss_ewis_perf_s Struct Reference

eWIS performance primitives. These data are assumed to be read once every sec. The counters holds increments compared to previous read. The namings and definitions are taken from ITU-T rec G.783.

```
#include <vtss_wis_api.h>
```

Data Fields

- u32 pn_ebc_s
- u32 pn_ebc_l
- u32 pf_ebc_l
- u32 pn_ebc_p
- u32 pf_ebc_p

7.54.1 Detailed Description

eWIS performance primitives. These data are assumed to be read once every sec. The counters holds increments compared to previous read. The namings and definitions are taken from ITU-T rec G.783.

Definition at line 176 of file vtss_wis_api.h.

7.54.2 Field Documentation

7.54.2.1 pn_ebc_s

```
u32 vtss_ewis_perf_s::pn_ebc_s
```

Section BIP error count

Definition at line 177 of file vtss_wis_api.h.

7.54.2.2 pn_ebc_l

```
u32 vtss_ewis_perf_s::pn_ebc_l
```

Near end line block (BIP) error count

Definition at line 178 of file vtss_wis_api.h.

7.54.2.3 pf_ebc_l

`u32 vtss_ewis_perf_s::pf_ebc_l`

Far end line block (BIP) error count

Definition at line 179 of file `vtss_wis_api.h`.

7.54.2.4 pn_ebc_p

`u32 vtss_ewis_perf_s::pn_ebc_p`

Path block error count

Definition at line 180 of file `vtss_wis_api.h`.

7.54.2.5 pf_ebc_p

`u32 vtss_ewis_perf_s::pf_ebc_p`

Far end path block error count

Definition at line 181 of file `vtss_wis_api.h`.

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

- `vtss_api/include/vtss_wis_api.h`

7.55 vtss_ewis_rdil_cons_act_s Struct Reference

eWIS RDI-L consequent actions

```
#include <vtss_wis_api.h>
```

Data Fields

- `BOOL rdil_on_los`
- `BOOL rdil_on_lof`
- `BOOL rdil_on_lopc`
- `BOOL rdil_on_ais_l`

7.55.1 Detailed Description

eWIS RDI-L consequent actions

Definition at line 83 of file vtss_wis_api.h.

7.55.2 Field Documentation

7.55.2.1 rdil_on_los

```
BOOL vtss_ewis_rdil_cons_act_s::rdil_on_los
```

TRUE = enable for RDI-L backreporting on LOS

Definition at line 84 of file vtss_wis_api.h.

7.55.2.2 rdil_on_lof

```
BOOL vtss_ewis_rdil_cons_act_s::rdil_on_lof
```

TRUE = enable for RDI-L backreporting on LOF

Definition at line 85 of file vtss_wis_api.h.

7.55.2.3 rdil_on_lopc

```
BOOL vtss_ewis_rdil_cons_act_s::rdil_on_lopc
```

TRUE = enable for RDI-L backreporting on LOPC, not supported in 8492

Definition at line 86 of file vtss_wis_api.h.

7.55.2.4 rdil_onais_l

```
BOOL vtss_ewis_rdil_cons_act_s::rdil_onais_l
```

TRUE = enable for RDI-L backreporting on AIS_L, not supported in 8492

Definition at line 87 of file vtss_wis_api.h.

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

- vtss_api/include/[vtss_wis_api.h](#)

7.56 vtss_ewis_sl_conf_s Struct Reference

signal label configuration

```
#include <vtss_wis_api.h>
```

Data Fields

- `u8 exsl`

7.56.1 Detailed Description

signal label configuration

Definition at line 306 of file vtss_wis_api.h.

7.56.2 Field Documentation

7.56.2.1 exsl

`u8 vtss_ewis_sl_conf_s::exsl`

expected signal label value

Definition at line 307 of file vtss_wis_api.h.

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

- vtss_api/include/[vtss_wis_api.h](#)

7.57 vtss_ewis_static_conf_s Struct Reference

eWIS static configuration data,

```
#include <vtss_wis_api.h>
```

Data Fields

- `u16 ewis_txctrl1`
- `u16 ewis_txctrl2`
- `u16 ewis_rx_ctrl1`
- `u16 ewis_mode_ctrl`
- `u16 ewis_tx_a1_a2`
- `u16 ewis_tx_c2_h1`
- `u16 ewis_tx_h2_h3`
- `u16 ewis_tx_z0_e1`
- `u16 ewis_rx_frm_ctrl1`
- `u16 ewis_rx_frm_ctrl2`
- `u16 ewis_lof_ctrl1`
- `u16 ewis_lof_ctrl2`
- `u16 ewis_rx_err_frc1`
- `u16 ewis_pmtick_ctrl`
- `u16 ewis_cnt_cfg`

7.57.1 Detailed Description

eWIS static configuration data,

Note

This is specific to 8487/8488-15 and should not be used for Daytona.

Definition at line 287 of file vtss_wis_api.h.

7.57.2 Field Documentation

7.57.2.1 `ewis_txctrl1`

`u16 vtss_ewis_static_conf_s::ewis_txctrl1`

WIS Vendor Specific Tx Control 1

Definition at line 288 of file vtss_wis_api.h.

7.57.2.2 `ewis_txctrl2`

`u16 vtss_ewis_static_conf_s::ewis_txctrl2`

WIS Vendor Specific Tx Control 2

Definition at line 289 of file vtss_wis_api.h.

7.57.2.3 ewis_rx_ctrl1

`u16 vtss_ewis_static_conf_s::ewis_rx_ctrl1`

E-WIS Rx Control 1

Definition at line 290 of file vtss_wis_api.h.

7.57.2.4 ewis_mode_ctrl

`u16 vtss_ewis_static_conf_s::ewis_mode_ctrl`

E-WIS Mode Control (incl expected C2 Path label)

Definition at line 291 of file vtss_wis_api.h.

7.57.2.5 ewis_tx_a1_a2

`u16 vtss_ewis_static_conf_s::ewis_tx_a1_a2`

E-WIS Tx A1/A2 Octets (frame alignment)

Definition at line 292 of file vtss_wis_api.h.

7.57.2.6 ewis_tx_c2_h1

`u16 vtss_ewis_static_conf_s::ewis_tx_c2_h1`

E-WIS Tx C2/H1 Octets

Definition at line 293 of file vtss_wis_api.h.

7.57.2.7 ewis_tx_h2_h3

`u16 vtss_ewis_static_conf_s::ewis_tx_h2_h3`

E-WIS Tx H2/H3 Octets

Definition at line 294 of file vtss_wis_api.h.

7.57.2.8 ewis_tx_z0_e1

`u16 vtss_ewis_static_conf_s::ewis_tx_z0_e1`

E-WIS Tx Z0/E1 Octets

Definition at line 295 of file `vtss_wis_api.h`.

7.57.2.9 ewis_rx_frm_ctrl1

`u16 vtss_ewis_static_conf_s::ewis_rx_frm_ctrl1`

E-WIS Rx Framer Control 1

Definition at line 296 of file `vtss_wis_api.h`.

7.57.2.10 ewis_rx_frm_ctrl2

`u16 vtss_ewis_static_conf_s::ewis_rx_frm_ctrl2`

E-WIS Rx Framer Control 2

Definition at line 297 of file `vtss_wis_api.h`.

7.57.2.11 ewis_lof_ctrl1

`u16 vtss_ewis_static_conf_s::ewis_lof_ctrl1`

E-WIS Loss of Frame Control 1

Definition at line 298 of file `vtss_wis_api.h`.

7.57.2.12 ewis_lof_ctrl2

`u16 vtss_ewis_static_conf_s::ewis_lof_ctrl2`

E-WIS Loss of Frame Control 2

Definition at line 299 of file `vtss_wis_api.h`.

7.57.2.13 ewis_rx_err_frc1

```
u16 vtss_ewis_static_conf_s::ewis_rx_err_frc1
```

E-WIS Rx Error Force Control 1 (incl RXLOF_ON_LOPC and APS_THRES configuration)

Definition at line 300 of file vtss_wis_api.h.

7.57.2.14 ewis_pmtick_ctrl

```
u16 vtss_ewis_static_conf_s::ewis_pmtick_ctrl
```

E-WIS Performance Monitor Control (define how PMTICK works)

Definition at line 301 of file vtss_wis_api.h.

7.57.2.15 ewis_cnt_cfg

```
u16 vtss_ewis_static_conf_s::ewis_cnt_cfg
```

E-WIS Counter Configuration (bit/block mode)

Definition at line 302 of file vtss_wis_api.h.

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

- vtss_api/include/vtss_wis_api.h

7.58 vtss_ewis_status_s Struct Reference

eWIS status

```
#include <vtss_wis_api.h>
```

Data Fields

- [BOOL fault](#)
- [BOOL link_stat](#)

7.58.1 Detailed Description

eWIS status

Definition at line 147 of file vtss_wis_api.h.

7.58.2 Field Documentation

7.58.2.1 fault

`BOOL vtss_ewis_status_s::fault`

Fault condition (Latch on high) i.e. = true if any fault has occurred since previous read

Definition at line 148 of file `vtss_wis_api.h`.

7.58.2.2 link_stat

`BOOL vtss_ewis_status_s::link_stat`

Link status condition (Latch on low) i.e. = false if any link error has occurred since previous read

Definition at line 149 of file `vtss_wis_api.h`.

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

- `vtss_api/include/vtss_wis_api.h`

7.59 `vtss_ewis_test_conf_s` Struct Reference

eWIS test configuration

```
#include <vtss_wis_api.h>
```

Data Fields

- `BOOL loopback`
- `vtss_ewis_test_pattern_t test_pattern_gen`
- `vtss_ewis_test_pattern_t test_pattern_ana`

7.59.1 Detailed Description

eWIS test configuration

Definition at line 206 of file `vtss_wis_api.h`.

7.59.2 Field Documentation

7.59.2.1 loopback

`BOOL vtss_ewis_test_conf_s::loopback`

loop output from Tx to Rx

Definition at line 207 of file vtss_wis_api.h.

7.59.2.2 test_pattern_gen

`vtss_ewis_test_pattern_t vtss_ewis_test_conf_s::test_pattern_gen`

test pattern generation configuration

Definition at line 208 of file vtss_wis_api.h.

7.59.2.3 test_pattern_ana

`vtss_ewis_test_pattern_t vtss_ewis_test_conf_s::test_pattern_ana`

test pattern analyzer configuration

Definition at line 209 of file vtss_wis_api.h.

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

- `vtss_api/include/vtss_wis_api.h`

7.60 vtss_ewis_test_status_s Struct Reference

eWIS test status

```
#include <vtss_wis_api.h>
```

Data Fields

- `u16 tstpat_cnt`
- `BOOL ana_sync`

7.60.1 Detailed Description

eWIS test status

Definition at line 213 of file vtss_wis_api.h.

7.60.2 Field Documentation

7.60.2.1 tstpat_cnt

```
u16 vtss_ewis_test_status_s::tstpat_cnt
```

PRBS31 test pattern error counter.

Definition at line 214 of file vtss_wis_api.h.

7.60.2.2 ana_sync

```
BOOL vtss_ewis_test_status_s::ana_sync
```

PRBS31 pattern checker is synchronized to the data.

Definition at line 215 of file vtss_wis_api.h.

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

- vtss_api/include/[vtss_wis_api.h](#)

7.61 vtss_ewis_tti_s Struct Reference

Trail Trace Identifier type.

```
#include <vtss_wis_api.h>
```

Data Fields

- [vtss_ewis_tti_mode_t mode](#)
- [u8 tti](#) [64]
- [BOOL valid](#)

7.61.1 Detailed Description

Trail Trace Identifier type.

Definition at line 55 of file vtss_wis_api.h.

7.61.2 Field Documentation

7.61.2.1 mode

`vtss_ewis_tti_mode_t vtss_ewis_tti_s::mode`

trace identifier mode (1,16,64 bytes)

Definition at line 56 of file vtss_wis_api.h.

7.61.2.2 tti

`u8 vtss_ewis_tti_s::tti[64]`

trace identifier value

Definition at line 57 of file vtss_wis_api.h.

7.61.2.3 valid

`BOOL vtss_ewis_tti_s::valid`

Identifies the Accepted valid TTI

Definition at line 58 of file vtss_wis_api.h.

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

- vtss_api/include/[vtss_wis_api.h](#)

7.62 vtss_ewis_tx_oh_s Struct Reference

WIS transmitted overhead data. only a few oh bytes can be set dynamically. These OH bytes are not configurable from the API: H4 : multiframe indicator M0/M1: STS-1 Line Remote Error Indication (REI) G1: Path status.

```
#include <vtss_wis_api.h>
```

Data Fields

- `u8 tx_dcc_s [3]`
- `u8 tx_e1`
- `u8 tx_f1`
- `u8 tx_z0`
- `u8 tx_dcc_l [9]`
- `u8 tx_e2`
- `u16 tx_k1_k2`
- `u8 tx_s1`
- `u16 tx_z1_z2`
- `u8 tx_c2`
- `u8 tx_f2`
- `u8 tx_n1`
- `u16 tx_z3_z4`

7.62.1 Detailed Description

WIS transmitted overhead data. only a few oh bytes can be set dynamically. These OH bytes are not configurable from the API: H4 : multiframe indicator M0/M1: STS-1 Line Remote Error Indication (REI) G1: Path status.

Definition at line 224 of file vtss_wis_api.h.

7.62.2 Field Documentation

7.62.2.1 tx_dcc_s

`u8 vtss_ewis_tx_oh_s::tx_dcc_s[3]`

< Section Overhead: Section Data Communications Channel(DCC) D1-D3

Definition at line 226 of file vtss_wis_api.h.

7.62.2.2 tx_e1

`u8 vtss_ewis_tx_oh_s::tx_e1`

Orderwire

Definition at line 227 of file vtss_wis_api.h.

7.62.2.3 tx_f1

`u8 vtss_ewis_tx_oh_s::tx_f1`

Section User Channel

Definition at line 228 of file vtss_wis_api.h.

7.62.2.4 tx_z0

`u8 vtss_ewis_tx_oh_s::tx_z0`

Reserved for Section growth line overhead:

Definition at line 229 of file vtss_wis_api.h.

7.62.2.5 tx_dcc_l

`u8 vtss_ewis_tx_oh_s::tx_dcc_l[9]`

Line Data Communications Channel (DCC) D4-D12

Definition at line 231 of file vtss_wis_api.h.

7.62.2.6 tx_e2

`u8 vtss_ewis_tx_oh_s::tx_e2`

Orderwire

Definition at line 232 of file vtss_wis_api.h.

7.62.2.7 tx_k1_k2

`u16 vtss_ewis_tx_oh_s::tx_k1_k2`

Automatic protection switch (APS) channel and Line Remote Defect Identifier (RDI-L)

Definition at line 233 of file vtss_wis_api.h.

7.62.2.8 tx_s1

`u8 vtss_ewis_tx_oh_s::tx_s1`

Synchronization messaging

Definition at line 234 of file vtss_wis_api.h.

7.62.2.9 tx_z1_z2

`u16 vtss_ewis_tx_oh_s::tx_z1_z2`

Reserved for Line growth path overhead:

Definition at line 235 of file vtss_wis_api.h.

7.62.2.10 tx_c2

`u8 vtss_ewis_tx_oh_s::tx_c2`

Transmitted C2 path label

Definition at line 237 of file vtss_wis_api.h.

7.62.2.11 tx_f2

`u8 vtss_ewis_tx_oh_s::tx_f2`

Path User Channel

Definition at line 238 of file vtss_wis_api.h.

7.62.2.12 tx_n1

`u8 vtss_ewis_tx_oh_s::tx_n1`

Tandem connection maintenance/Path data channel

Definition at line 239 of file vtss_wis_api.h.

7.62.2.13 tx_z3_z4

u16 vtss_ewis_tx_oh_s::tx_z3_z4

Reserved for Path growth

Definition at line 240 of file vtss_wis_api.h.

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

- vtss_api/include/[vtss_wis_api.h](#)

7.63 vtss_ewis_tx_passthru_s Struct Reference

eWIS overhead passthru configuration.

```
#include <vtss_wis_api.h>
```

Data Fields

- BOOL tx_j0
- BOOL tx_z0
- BOOL tx_b1
- BOOL tx_e1
- BOOL tx_f1
- BOOL tx_dcc_s
- BOOL tx_soh
- BOOL tx_b2
- BOOL tx_k1
- BOOL tx_k2
- BOOL tx_reil
- BOOL tx_dcc_l
- BOOL tx_s1
- BOOL tx_e2
- BOOL tx_z1_z2
- BOOL tx_loh

7.63.1 Detailed Description

eWIS overhead passthru configuration.

Definition at line 245 of file vtss_wis_api.h.

7.63.2 Field Documentation

7.63.2.1 tx_j0

`BOOL vtss_ewis_tx_passthru_s::tx_j0`

< Section Overhead: j0 Section TTI passthrough

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

7.63.2.2 tx_z0

`BOOL vtss_ewis_tx_passthru_s::tx_z0`

z0 Section growth passthrough

Definition at line 248 of file `vtss_wis_api.h`.

7.63.2.3 tx_b1

`BOOL vtss_ewis_tx_passthru_s::tx_b1`

b1 BIP passthrough

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

7.63.2.4 tx_e1

`BOOL vtss_ewis_tx_passthru_s::tx_e1`

e1 order wire passthrough

Definition at line 250 of file `vtss_wis_api.h`.

7.63.2.5 tx_f1

`BOOL vtss_ewis_tx_passthru_s::tx_f1`

f1 Section user channel

Definition at line 251 of file `vtss_wis_api.h`.

7.63.2.6 tx_dcc_s

`BOOL vtss_ewis_tx_passthru_s::tx_dcc_s`

Section Data communication channel passthrough D1-D3

Definition at line 252 of file vtss_wis_api.h.

7.63.2.7 tx_soh

`BOOL vtss_ewis_tx_passthru_s::tx_soh`

Section Reserved National and unused bytes passthrough line overhead:

Definition at line 253 of file vtss_wis_api.h.

7.63.2.8 tx_b2

`BOOL vtss_ewis_tx_passthru_s::tx_b2`

b2 BIP passthrough

Definition at line 256 of file vtss_wis_api.h.

7.63.2.9 tx_k1

`BOOL vtss_ewis_tx_passthru_s::tx_k1`

k1 passthrough

Definition at line 257 of file vtss_wis_api.h.

7.63.2.10 tx_k2

`BOOL vtss_ewis_tx_passthru_s::tx_k2`

k2 passthrough

Definition at line 258 of file vtss_wis_api.h.

7.63.2.11 tx_reil

`BOOL vtss_ewis_tx_passthru_s::tx_reil`

reil passthrough

Definition at line 259 of file vtss_wis_api.h.

7.63.2.12 tx_dcc_l

`BOOL vtss_ewis_tx_passthru_s::tx_dcc_l`

Section Data communication channel passthrough D4-D12

Definition at line 260 of file vtss_wis_api.h.

7.63.2.13 tx_s1

`BOOL vtss_ewis_tx_passthru_s::tx_s1`

Synchronization messaging passthrough

Definition at line 261 of file vtss_wis_api.h.

7.63.2.14 tx_e2

`BOOL vtss_ewis_tx_passthru_s::tx_e2`

order wire passthrough

Definition at line 262 of file vtss_wis_api.h.

7.63.2.15 tx_z1_z2

`BOOL vtss_ewis_tx_passthru_s::tx_z1_z2`

Reserved for path growth passthrough

Definition at line 263 of file vtss_wis_api.h.

7.63.2.16 tx_loh

`BOOL vtss_ewis_tx_passthru_s::tx_loh`

Line Reserved National and unused bytes passthrough

Definition at line 264 of file vtss_wis_api.h.

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

- vtss_api/include/[vtss_wis_api.h](#)

7.64 vtss_fan_conf_t Struct Reference

Fan specifications.

```
#include <vtss_misc_api.h>
```

Data Fields

- [vtss_fan_pwd_freq_t](#) fan_pwm_freq
- `BOOL` fan_low_pol
- `BOOL` fan_open_col
- [vtss_fan_type_t](#) type
- `u32` ppr

7.64.1 Detailed Description

Fan specifications.

Definition at line 1148 of file vtss_misc_api.h.

7.64.2 Field Documentation

7.64.2.1 fan_pwm_freq

`vtss_fan_pwd_freq_t vtss_fan_conf_t::fan_pwm_freq`

Fan PWM frequency

Definition at line 1150 of file vtss_misc_api.h.

7.64.2.2 fan_low_pol

`BOOL vtss_fan_conf_t::fan_low_pol`

Fan polarity of the PWM output. TRUE = PWM is logic 0 when on. FALSE = PWM is logic 1 when on

Definition at line 1151 of file `vtss_misc_api.h`.

7.64.2.3 fan_open_col

`BOOL vtss_fan_conf_t::fan_open_col`

PWM output is open collector if TRUE.

Definition at line 1152 of file `vtss_misc_api.h`.

7.64.2.4 type

`vtss_fan_type_t vtss_fan_conf_t::type`

2,3 or 4 wire fan type

Definition at line 1153 of file `vtss_misc_api.h`.

7.64.2.5 ppr

`u32 vtss_fan_conf_t::ppr`

Pulses per rotation. Only valid for 3 and 4 wire fans

Definition at line 1157 of file `vtss_misc_api.h`.

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

- `vtss_api/include/vtss_misc_api.h`

7.65 vtss_gpio_10g_gpio_mode_t Struct Reference

GPIO configured mode.

```
#include <vtss_phy_10g_api.h>
```

Data Fields

- `vtss_10g_phy_gpio_t mode`
- `vtss_port_no_t port`
- `vtss_gpio_10g_input_t input`
- `vtss_gpio_10g_gpio_intr_sgnl_t in_sig`
- `vtss_gpio_10g_no_t p_gpio`
- `vtss_gpio_10g_chan_intrpt_t c_intrpt`
- `u16 source`
- `u32 agrr_intrpt`
- `BOOL use_as_intrpt`
- `BOOL p_gpio_intrpt`

7.65.1 Detailed Description

GPIO configured mode.

GPIO input modes

Definition at line 2480 of file vtss_phy_10g_api.h.

7.65.2 Field Documentation

7.65.2.1 mode

`vtss_10g_phy_gpio_t vtss_gpio_10g_gpio_mode_t::mode`

Mode of this GPIO pin

Definition at line 2482 of file vtss_phy_10g_api.h.

7.65.2.2 port

`vtss_port_no_t vtss_gpio_10g_gpio_mode_t::port`

In case of VTSS_10G_PHY_GPIO_WIS_INT mode, this is the interrupt port number that is related to this GPIO In case of VTSS_10G_PHY_GPIO_PCS_RX_FAULT mode, this is the PCS status port number that is related to this GPIO

Definition at line 2483 of file vtss_phy_10g_api.h.

7.65.2.3 input

`vtss_gpio_10g_input_t` `vtss_gpio_10g_gpio_mode_t::input`

GPIO input modes

Definition at line 2485 of file vtss_phy_10g_api.h.

7.65.2.4 in_sig

`vtss_gpio_10g_gpio_intr_sgnl_t` `vtss_gpio_10g_gpio_mode_t::in_sig`

Internal signal that to be routed through GPIO

Definition at line 2486 of file vtss_phy_10g_api.h.

7.65.2.5 p_gpio

`vtss_gpio_10g_no_t` `vtss_gpio_10g_gpio_mode_t::p_gpio`

Per channel GPIO number

Definition at line 2487 of file vtss_phy_10g_api.h.

7.65.2.6 c_intrpt

`vtss_gpio_10g_chan_intrpt_t` `vtss_gpio_10g_gpio_mode_t::c_intrpt`

Per Channel interrupt,

Definition at line 2488 of file vtss_phy_10g_api.h.

7.65.2.7 source

`u16` `vtss_gpio_10g_gpio_mode_t::source`

source of GPIO, approriate value from GPIO_OUT_CFG_X register field GPIO_X_SEL

Definition at line 2489 of file vtss_phy_10g_api.h.

7.65.2.8 aggr_intrpt

`u32 vtss_gpio_10g_gpio_mode_t::aggr_intrpt`

Bitmask corresponds to aggregated interrupt

Definition at line 2490 of file vtss_phy_10g_api.h.

7.65.2.9 use_as_intrpt

`BOOL vtss_gpio_10g_gpio_mode_t::use_as_intrpt`

Change in GPIO generates GPIO interrupt ,supported on MALIBU

Definition at line 2491 of file vtss_phy_10g_api.h.

7.65.2.10 p_gpio_intrpt

`BOOL vtss_gpio_10g_gpio_mode_t::p_gpio_intrpt`

Port GPIO Change interrupt

Definition at line 2492 of file vtss_phy_10g_api.h.

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

- [vtss_api/include/vtss_phy_10g_api.h](#)

7.66 vtss_init_conf_t Struct Reference

Initialization configuration.

```
#include <vtss_init_api.h>
```

Data Fields

- [vtss_reg_read_t reg_read](#)
- [vtss_reg_write_t reg_write](#)
- [vtss_miim_read_t miim_read](#)
- [vtss_miim_write_t miim_write](#)
- [vtss_mmd_read_t mmd_read](#)
- [vtss_mmd_read_inc_t mmd_read_inc](#)
- [vtss_mmd_write_t mmd_write](#)
- [vtss_spi_read_write_t spi_read_write](#)
- [vtss_spi_32bit_read_write_t spi_32bit_read_write](#)
- [vtss_spi_64bit_read_write_t spi_64bit_read_write](#)
- [BOOL warm_start_enable](#)
- [vtss_restart_info_src_t restart_info_src](#)
- [vtss_port_no_t restart_info_port](#)
- [vtss_port_mux_mode_t mux_mode](#)
- [vtss_pi_conf_t pi](#)
- [vtss_serdes_macro_conf_t serdes](#)
- [vtss_qs_conf_t qs_conf](#)

7.66.1 Detailed Description

Initialization configuration.

Definition at line 515 of file vtss_init_api.h.

7.66.2 Field Documentation

7.66.2.1 reg_read

```
vtss_reg_read_t vtss_init_conf_t::reg_read
```

Register read function

Definition at line 517 of file vtss_init_api.h.

7.66.2.2 reg_write

```
vtss_reg_write_t vtss_init_conf_t::reg_write
```

Register write function

Definition at line 518 of file vtss_init_api.h.

7.66.2.3 miim_read

```
vtss_miim_read_t vtss_init_conf_t::miim_read
```

MII management read function

Definition at line 521 of file vtss_init_api.h.

7.66.2.4 miim_write

```
vtss_miim_write_t vtss_init_conf_t::miim_write
```

MII management write function

Definition at line 522 of file vtss_init_api.h.

7.66.2.5 mmd_read

`vtss_mmd_read_t` `vtss_init_conf_t::mmd_read`

MMD management read function

Definition at line 525 of file vtss_init_api.h.

7.66.2.6 mmd_read_inc

`vtss_mmd_read_inc_t` `vtss_init_conf_t::mmd_read_inc`

MMD management read increment function

Definition at line 526 of file vtss_init_api.h.

7.66.2.7 mmd_write

`vtss_mmd_write_t` `vtss_init_conf_t::mmd_write`

MMD management write function

Definition at line 527 of file vtss_init_api.h.

7.66.2.8 spi_read_write

`vtss_spi_read_write_t` `vtss_init_conf_t::spi_read_write`

Board specific SPI read/write callout function

Definition at line 529 of file vtss_init_api.h.

7.66.2.9 spi_32bit_read_write

`vtss_spi_32bit_read_write_t` `vtss_init_conf_t::spi_32bit_read_write`

Board specific SPI read/write callout function for 32 bit data

Definition at line 531 of file vtss_init_api.h.

7.66.2.10 spi_64bit_read_write

`vtss_spi_64bit_read_write_t vtss_init_conf_t::spi_64bit_read_write`

Board specific SPI read/write callout function for 64 bit data

Definition at line 532 of file vtss_init_api.h.

7.66.2.11 warm_start_enable

`BOOL vtss_init_conf_t::warm_start_enable`

Allow warm start

Definition at line 535 of file vtss_init_api.h.

7.66.2.12 restart_info_src

`vtss_restart_info_src_t vtss_init_conf_t::restart_info_src`

Source of restart information

Definition at line 536 of file vtss_init_api.h.

7.66.2.13 restart_info_port

`vtss_port_no_t vtss_init_conf_t::restart_info_port`

Port used to store PHY restart information

Definition at line 537 of file vtss_init_api.h.

7.66.2.14 mux_mode

`vtss_port_mux_mode_t vtss_init_conf_t::mux_mode`

Mux mode (port connection to Serdes Macros)

Definition at line 541 of file vtss_init_api.h.

7.66.2.15 pi

```
vtss_pi_conf_t vtss_init_conf_t::pi
```

Parallel Interface configuration

Definition at line 547 of file vtss_init_api.h.

7.66.2.16 serdes

```
vtss_serdes_macro_conf_t vtss_init_conf_t::serdes
```

Serdes macro configuration

Definition at line 550 of file vtss_init_api.h.

7.66.2.17 qs_conf

```
vtss_qs_conf_t vtss_init_conf_t::qs_conf
```

Queue system configuration

Definition at line 559 of file vtss_init_api.h.

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

- vtss_api/include/[vtss_init_api.h](#)

7.67 vtss_inst_create_t Struct Reference

Create structure.

```
#include <vtss_init_api.h>
```

Data Fields

- [vtss_target_type_t target](#)

7.67.1 Detailed Description

Create structure.

Definition at line 78 of file vtss_init_api.h.

7.67.2 Field Documentation

7.67.2.1 target

```
vtss_target_type_t vtss_inst_create_t::target
```

Target type

Definition at line 79 of file vtss_init_api.h.

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

- [vtss_api/include/vtss_init_api.h](#)

7.68 vtss_ip_addr_t Struct Reference

Either an IPv4 or IPv6 address.

```
#include <types.h>
```

Data Fields

- [vtss_ip_type_t type](#)
- union {
 - [vtss_ipv4_t ipv4](#)
 - [vtss_ipv6_t ipv6](#)}
- [addr](#)

7.68.1 Detailed Description

Either an IPv4 or IPv6 address.

Definition at line 813 of file types.h.

7.68.2 Field Documentation

7.68.2.1 type

```
vtss_ip_type_t vtss_ip_addr_t::type
```

Union type

Definition at line 814 of file types.h.

7.68.2.2 ipv4

`vtss_ipv4_t vtss_ip_addr_t::ipv4`

IPv4 address

Definition at line 816 of file types.h.

7.68.2.3 ipv6

`vtss_ipv6_t vtss_ip_addr_t::ipv6`

IPv6 address

Definition at line 817 of file types.h.

7.68.2.4 addr

`union { ... } vtss_ip_addr_t::addr`

IP address

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

- [vtss_api/include/vtss/api/types.h](#)

7.69 vtss_ip_network_t Struct Reference

IPv6 network.

```
#include <types.h>
```

Data Fields

- [vtss_ip_addr_t address](#)
- [vtss_prefix_size_t prefix_size](#)

7.69.1 Detailed Description

IPv6 network.

Definition at line 836 of file types.h.

7.69.2 Field Documentation

7.69.2.1 address

`vtss_ip_addr_t vtss_ip_network_t::address`

Network address

Definition at line 838 of file types.h.

7.69.2.2 prefix_size

`vtss_prefix_size_t vtss_ip_network_t::prefix_size`

Prefix size

Definition at line 839 of file types.h.

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

- `vtss_api/include/vtss/api/types.h`

7.70 vtss_ipv4_network_t Struct Reference

IPv4 network.

```
#include <types.h>
```

Data Fields

- `vtss_ipv4_t address`
- `vtss_prefix_size_t prefix_size`

7.70.1 Detailed Description

IPv4 network.

Definition at line 822 of file types.h.

7.70.2 Field Documentation

7.70.2.1 address

`vtss_ipv4_t vtss_ipv4_network_t::address`

Network address

Definition at line 824 of file types.h.

7.70.2.2 prefix_size

`vtss_prefix_size_t vtss_ipv4_network_t::prefix_size`

Prefix size

Definition at line 825 of file types.h.

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

- `vtss_api/include/vtss/api/types.h`

7.71 **vtss_ipv4_uc_t** Struct Reference

IPv4 unicast routing entry.

```
#include <types.h>
```

Data Fields

- `vtss_ipv4_network_t network`
- `vtss_ipv4_t destination`

7.71.1 Detailed Description

IPv4 unicast routing entry.

Definition at line 852 of file types.h.

7.71.2 Field Documentation

7.71.2.1 network

```
vtss_ipv4_network_t vtss_ipv4_uc_t::network
```

Network to route

Definition at line 854 of file types.h.

7.71.2.2 destination

```
vtss_ipv4_t vtss_ipv4_uc_t::destination
```

IP address of next-hop router. Zero if local route

Definition at line 855 of file types.h.

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

- [vtss_api/include/vtss/api/types.h](#)

7.72 vtss_ipv6_network_t Struct Reference

IPv6 network.

```
#include <types.h>
```

Data Fields

- [vtss_ipv6_t address](#)
- [vtss_prefix_size_t prefix_size](#)

7.72.1 Detailed Description

IPv6 network.

Definition at line 829 of file types.h.

7.72.2 Field Documentation

7.72.2.1 address

`vtss_ipv6_t vtss_ipv6_network_t::address`

Network address

Definition at line 831 of file types.h.

7.72.2.2 prefix_size

`vtss_prefix_size_t vtss_ipv6_network_t::prefix_size`

Prefix size

Definition at line 832 of file types.h.

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

- `vtss_api/include/vtss/api/types.h`

7.73 vtss_ipv6_t Struct Reference

IPv6 address/mask.

```
#include <types.h>
```

Data Fields

- `u8 addr [16]`

7.73.1 Detailed Description

IPv6 address/mask.

Definition at line 797 of file types.h.

7.73.2 Field Documentation

7.73.2.1 addr

```
u8 vtss_ipv6_t::addr[16]
```

Address

Definition at line 799 of file types.h.

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

- [vtss_api/include/vtss/api/types.h](#)

7.74 vtss_ipv6_uc_t Struct Reference

IPv6 routing entry.

```
#include <types.h>
```

Data Fields

- [vtss_ipv6_network_t network](#)
- [vtss_ipv6_t destination](#)

7.74.1 Detailed Description

IPv6 routing entry.

Definition at line 860 of file types.h.

7.74.2 Field Documentation

7.74.2.1 network

```
vtss_ipv6_network_t vtss_ipv6_uc_t::network
```

Network to route

Definition at line 862 of file types.h.

7.74.2.2 destination

`vtss_ipv6_t vtss_ipv6_uc_t::destination`

IP address of next-hop router. Zero if local route

Definition at line 863 of file types.h.

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

- `vtss_api/include/vtss/api/types.h`

7.75 vtss_irq_conf_t Struct Reference

Interrupt configuration options.

```
#include <vtss_misc_api.h>
```

Data Fields

- `BOOL external`
- `u8 destination`

7.75.1 Detailed Description

Interrupt configuration options.

Definition at line 972 of file vtss_misc_api.h.

7.75.2 Field Documentation

7.75.2.1 external

`BOOL vtss_irq_conf_t::external`

Redirect to external IRQ

Definition at line 973 of file vtss_misc_api.h.

7.75.2.2 destination

`u8 vtss_irq_conf_t::destination`

IRQ destination index

Definition at line 974 of file vtss_misc_api.h.

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

- `vtss_api/include/vtss_misc_api.h`

7.76 vtss_irq_status_t Struct Reference

Interrupt status structure.

```
#include <vtss_misc_api.h>
```

Data Fields

- `u32 active`
- `u32 raw_ident`
- `u32 raw_status`
- `u32 raw_mask`

7.76.1 Detailed Description

Interrupt status structure.

Definition at line 980 of file vtss_misc_api.h.

7.76.2 Field Documentation

7.76.2.1 active

`u32 vtss_irq_status_t::active`

Bitmap for pending IRQs (VTSS_IRQ_xxx)

Definition at line 981 of file vtss_misc_api.h.

7.76.2.2 raw_ident

`u32 vtss_irq_status_t::raw_ident`

RAW (target dependentant) bitmap for active pending IRQs

Definition at line 982 of file vtss_misc_api.h.

7.76.2.3 raw_status

`u32 vtss_irq_status_t::raw_status`

RAW (target dependentant) bitmap for all pending IRQs

Definition at line 983 of file vtss_misc_api.h.

7.76.2.4 raw_mask

`u32 vtss_irq_status_t::raw_mask`

RAW (target dependentant) bitmap for IRQs mask

Definition at line 984 of file vtss_misc_api.h.

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

- vtss_api/include/vtss_misc_api.h

7.77 vtss_l3_common_conf_t Struct Reference

Common configurations for all routing legs.

```
#include <vtss_l3_api.h>
```

Data Fields

- `vtss_l3_rleg_common_mode_t rleg_mode`
- `vtss_mac_t base_address`
- `BOOL routing_enable`

7.77.1 Detailed Description

Common configurations for all routing legs.

Definition at line 163 of file vtss_l3_api.h.

7.77.2 Field Documentation

7.77.2.1 rleg_mode

`vtss_l3_rleg_common_mode_t` `vtss_l3_common_conf_t::rleg_mode`

Common rleg-mode for all routing legs.

Definition at line 166 of file vtss_l3_api.h.

7.77.2.2 base_address

`vtss_mac_t` `vtss_l3_common_conf_t::base_address`

Base mac address used to derive addresses for all routing legs.

Definition at line 169 of file vtss_l3_api.h.

7.77.2.3 routing_enable

`BOOL` `vtss_l3_common_conf_t::routing_enable`

Globally enable/disable routing.

Definition at line 172 of file vtss_l3_api.h.

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

- `vtss_api/include/vtss_l3_api.h`

7.78 vtss_l3_counters_t Struct Reference

Routing interface statics counter.

```
#include <types.h>
```

Data Fields

- `u64 ipv4uc_received_octets`
- `u64 ipv4uc_received_frames`
- `u64 ipv6uc_received_octets`
- `u64 ipv6uc_received_frames`
- `u64 ipv4uc_transmitted_octets`
- `u64 ipv4uc_transmitted_frames`
- `u64 ipv6uc_transmitted_octets`
- `u64 ipv6uc_transmitted_frames`

7.78.1 Detailed Description

Routing interface statics counter.

Definition at line 886 of file types.h.

7.78.2 Field Documentation

7.78.2.1 ipv4uc_received_octets

```
u64 vtss_l3_counters_t::ipv4uc_received_octets
```

IPv4UC octets received and hardware forwarded

Definition at line 887 of file types.h.

7.78.2.2 ipv4uc_received_frames

```
u64 vtss_l3_counters_t::ipv4uc_received_frames
```

IPv4UC frames received and hardware forwarded

Definition at line 888 of file types.h.

7.78.2.3 ipv6uc_received_octets

```
u64 vtss_l3_counters_t::ipv6uc_received_octets
```

IPv6UC octets received and hardware forwarded

Definition at line 889 of file types.h.

7.78.2.4 ipv6uc_received_frames

```
u64 vtss_l3_counters_t::ipv6uc_received_frames
```

IPv6UC frames received and hardware forwarded

Definition at line 890 of file types.h.

7.78.2.5 ipv4uc_transmitted_octets

`u64 vtss_l3_counters_t::ipv4uc_transmitted_octets`

IPv4UC octets transmitted

Definition at line 892 of file types.h.

7.78.2.6 ipv4uc_transmitted_frames

`u64 vtss_l3_counters_t::ipv4uc_transmitted_frames`

IPv4UC frames transmitted

Definition at line 893 of file types.h.

7.78.2.7 ipv6uc_transmitted_octets

`u64 vtss_l3_counters_t::ipv6uc_transmitted_octets`

IPv6UC octets transmitted

Definition at line 894 of file types.h.

7.78.2.8 ipv6uc_transmitted_frames

`u64 vtss_l3_counters_t::ipv6uc_transmitted_frames`

IPv6UC frames transmitted

Definition at line 895 of file types.h.

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

- `vtss_api/include/vtss/api/types.h`

7.79 vtss_l3_neighbour_t Struct Reference

Neighbour entry.

```
#include <vtss_l3_api.h>
```

Data Fields

- `vtss_mac_t dmac`
- `vtss_vid_t vlan`
- `vtss_ip_addr_t dip`

7.79.1 Detailed Description

Neighbour entry.

Definition at line 230 of file `vtss_l3_api.h`.

7.79.2 Field Documentation

7.79.2.1 dmac

`vtss_mac_t vtss_l3_neighbour_t::dmac`

MAC address of destination

Definition at line 233 of file `vtss_l3_api.h`.

7.79.2.2 vlan

`vtss_vid_t vtss_l3_neighbour_t::vlan`

VLAN of destination

Definition at line 236 of file `vtss_l3_api.h`.

7.79.2.3 dip

`vtss_ip_addr_t vtss_l3_neighbour_t::dip`

IP address of destination

Definition at line 239 of file `vtss_l3_api.h`.

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

- `vtss_api/include/vtss_l3_api.h`

7.80 vtss_l3_rleg_conf_t Struct Reference

Router leg control structure.

```
#include <vtss_l3_api.h>
```

Data Fields

- `BOOL ipv4_unicast_enable`
- `BOOL ipv6_unicast_enable`
- `BOOL ipv4_icmp_redirect_enable`
- `BOOL ipv6_icmp_redirect_enable`
- `vtss_vid_t vlan`
- `BOOL vrid0_enable`
- `vtss_l3_vrid_t vrid0`
- `BOOL vrid1_enable`
- `vtss_l3_vrid_t vrid1`

7.80.1 Detailed Description

Router leg control structure.

Definition at line 176 of file `vtss_l3_api.h`.

7.80.2 Field Documentation

7.80.2.1 ipv4_unicast_enable

```
BOOL vtss_l3_rleg_conf_t::ipv4_unicast_enable
```

Enable IPv4 unicast routing

Definition at line 179 of file `vtss_l3_api.h`.

7.80.2.2 ipv6_unicast_enable

```
BOOL vtss_l3_rleg_conf_t::ipv6_unicast_enable
```

Enable IPv6 unicast routing

Definition at line 182 of file `vtss_l3_api.h`.

7.80.2.3 ipv4_icmp_redirect_enable

`BOOL vtss_l3_rleg_conf_t::ipv4_icmp_redirect_enable`

Enable IPv4 icmp redirect

Definition at line 185 of file vtss_l3_api.h.

7.80.2.4 ipv6_icmp_redirect_enable

`BOOL vtss_l3_rleg_conf_t::ipv6_icmp_redirect_enable`

Enable IPv6 icmp redirect

Definition at line 188 of file vtss_l3_api.h.

7.80.2.5 vlan

`vtss_vid_t vtss_l3_rleg_conf_t::vlan`

Vlan for which the router leg is instantiated

Definition at line 191 of file vtss_l3_api.h.

7.80.2.6 vrid0_enable

`BOOL vtss_l3_rleg_conf_t::vrid0_enable`

Enable/disable VRRP for a given router leg.

The hardware has support for enabling 0-2 VRID's for a given router leg. This is activated by configured vrid0_enable and vrid1_enable. The actual VRID the route is assigned to is configured in vrid0/vrid1.

JR1-NOTE: When enabling vrrp for JR1-revb, the hardware will consider all destination mac addresses matching 00-00-5E-00-XX-{VRID} as VRRP address. For versions earlier than rev-B VRRP is only supported for IPv4 (00-00-5E-00-01-{VRID}).

Definition at line 204 of file vtss_l3_api.h.

7.80.2.7 vrid0

```
vtss_l3_vrid_t vtss_l3_rleg_conf_t::vrid0
```

The VRID value assigned to this router leg.

Definition at line 207 of file vtss_l3_api.h.

7.80.2.8 vrid1_enable

```
BOOL vtss_l3_rleg_conf_t::vrid1_enable
```

Enable/disable vrid1 for this router leg.

Definition at line 210 of file vtss_l3_api.h.

7.80.2.9 vrid1

```
vtss_l3_vrid_t vtss_l3_rleg_conf_t::vrid1
```

The VRID value assigned to this router leg.

Definition at line 213 of file vtss_l3_api.h.

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

- vtss_api/include/[vtss_l3_api.h](#)

7.81 vtss_lcpll_status_t Struct Reference

Structure for Get PHY LC-PLL status.

```
#include <vtss_phy_api.h>
```

Data Fields

- [u8 lock_status](#)
- [u8 cal_done](#)
- [u8 cal_error](#)
- [u8 fsm_lock](#)
- [u8 fsm_stat](#)
- [u8 gain_stat](#)

7.81.1 Detailed Description

Structure for Get PHY LC-PLL status.

Definition at line 1777 of file vtss_phy_api.h.

7.81.2 Field Documentation

7.81.2.1 lock_status

```
u8 vtss_lcppll_status_t::lock_status
```

PLL lock status

Definition at line 1778 of file vtss_phy_api.h.

7.81.2.2 cal_done

```
u8 vtss_lcppll_status_t::cal_done
```

Calibration status

Definition at line 1779 of file vtss_phy_api.h.

7.81.2.3 cal_error

```
u8 vtss_lcppll_status_t::cal_error
```

Calibration Error indication

Definition at line 1780 of file vtss_phy_api.h.

7.81.2.4 fsm_lock

```
u8 vtss_lcppll_status_t::fsm_lock
```

FSM lock status

Definition at line 1781 of file vtss_phy_api.h.

7.81.2.5 fsm_stat

`u8 vtss_lcp11_status_t::fsm_stat`

FSM internal status

Definition at line 1782 of file vtss_phy_api.h.

7.81.2.6 gain_stat

`u8 vtss_lcp11_status_t::gain_stat`

VCO frequency step stop

Definition at line 1783 of file vtss_phy_api.h.

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

- `vtss_api/include/vtss_phy_api.h`

7.82 vtss_learn_mode_t Struct Reference

Learning mode.

```
#include <vtss_l2_api.h>
```

Data Fields

- `BOOL automatic`
- `BOOL cpu`
- `BOOL discard`

7.82.1 Detailed Description

Learning mode.

Definition at line 379 of file vtss_l2_api.h.

7.82.2 Field Documentation

7.82.2.1 automatic

`BOOL vtss_learn_mode_t::automatic`

Automatic learning done by switch chip (default enabled)

Definition at line 381 of file vtss_l2_api.h.

7.82.2.2 cpu

`BOOL vtss_learn_mode_t::cpu`

Learn frames copied to CPU (default disabled)

Definition at line 382 of file vtss_l2_api.h.

7.82.2.3 discard

`BOOL vtss_learn_mode_t::discard`

Learn frames discarded (default disabled)

Definition at line 383 of file vtss_l2_api.h.

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

- `vtss_api/include/vtss_l2_api.h`

7.83 vtss_mac_t Struct Reference

MAC Address.

```
#include <types.h>
```

Data Fields

- `u8 addr [6]`

7.83.1 Detailed Description

MAC Address.

Definition at line 643 of file types.h.

7.83.2 Field Documentation

7.83.2.1 addr

```
u8 vtss_mac_t::addr[6]
```

Network byte order

Definition at line 645 of file types.h.

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

- [vtss_api/include/vtss/api/types.h](#)

7.84 vtss_mac_table_entry_t Struct Reference

MAC address entry.

```
#include <vtss_l2_api.h>
```

Data Fields

- [vtss_vid_mac_t vid_mac](#)
- [BOOL destination \[VTSS_PORT_ARRAY_SIZE\]](#)
- [BOOL copy_to_cpu](#)
- [BOOL locked](#)
- [BOOL aged](#)
- [vtss_packet_rx_queue_t cpu_queue](#)
- struct {
 - [BOOL enable](#)
 - [BOOL remote_entry](#)
 - [vtss_vstax_upsid_t upsid](#)
 - [vtss_vstax_upspn_t upspn](#)} [vstax2](#)

7.84.1 Detailed Description

MAC address entry.

Definition at line 119 of file vtss_l2_api.h.

7.84.2 Field Documentation

7.84.2.1 vid_mac

`vtss_vid_mac_t vtss_mac_table_entry_t::vid_mac`

VLAN ID and MAC addr

Definition at line 121 of file vtss_l2_api.h.

7.84.2.2 destination

`BOOL vtss_mac_table_entry_t::destination[VTSS_PORT_ARRAY_SIZE]`

Dest. ports

Definition at line 122 of file vtss_l2_api.h.

7.84.2.3 copy_to_cpu

`BOOL vtss_mac_table_entry_t::copy_to_cpu`

CPU copy flag

Definition at line 123 of file vtss_l2_api.h.

7.84.2.4 locked

`BOOL vtss_mac_table_entry_t::locked`

Locked/static flag

Definition at line 124 of file vtss_l2_api.h.

7.84.2.5 aged

`BOOL vtss_mac_table_entry_t::aged`

Age flag

Definition at line 125 of file vtss_l2_api.h.

7.84.2.6 cpu_queue

`vtss_packet_rx_queue_t` `vtss_mac_table_entry_t::cpu_queue`

CPU queue

Definition at line 127 of file vtss_l2_api.h.

7.84.2.7 enable

`BOOL` `vtss_mac_table_entry_t::enable`

Use (UPSID, UPSPN) when adding entry

Definition at line 132 of file vtss_l2_api.h.

7.84.2.8 remote_entry

`BOOL` `vtss_mac_table_entry_t::remote_entry`

Local or remote entry when getting entry

Definition at line 133 of file vtss_l2_api.h.

7.84.2.9 upsid

`vtss_vstax_upsid_t` `vtss_mac_table_entry_t::upsid`

UPS identifier

Definition at line 134 of file vtss_l2_api.h.

7.84.2.10 upspn

`vtss_vstax_upspn_t` `vtss_mac_table_entry_t::upspn`

Logical port within UPS

Definition at line 135 of file vtss_l2_api.h.

7.84.2.11 vstax2

```
struct { ... } vtss_mac_table_entry_t::vstax2
```

Unit/port identification

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

- [vtss_api/include/vtss_l2_api.h](#)

7.85 vtss_mac_table_status_t Struct Reference

MAC address table status.

```
#include <vtss_l2_api.h>
```

Data Fields

- [vtss_event_t learned](#)
- [vtss_event_t replaced](#)
- [vtss_event_t moved](#)
- [vtss_event_t aged](#)

7.85.1 Detailed Description

MAC address table status.

Definition at line 358 of file [vtss_l2_api.h](#).

7.85.2 Field Documentation

7.85.2.1 learned

```
vtss_event_t vtss_mac_table_status_t::learned
```

One or more entries were learned

Definition at line 360 of file [vtss_l2_api.h](#).

7.85.2.2 replaced

`vtss_event_t vtss_mac_table_status_t::replaced`

One or more entries were replaced

Definition at line 361 of file `vtss_l2_api.h`.

7.85.2.3 moved

`vtss_event_t vtss_mac_table_status_t::moved`

One or more entries moved to another port

Definition at line 362 of file `vtss_l2_api.h`.

7.85.2.4 aged

`vtss_event_t vtss_mac_table_status_t::aged`

One or more entries were aged

Definition at line 363 of file `vtss_l2_api.h`.

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

- `vtss_api/include/vtss_l2_api.h`

7.86 vtss_mirror_conf_t Struct Reference

Mirror configuration.

```
#include <vtss_l2_api.h>
```

Data Fields

- `vtss_port_no_t port_no`
- `BOOL fwd_enable`
- `vtss_mirror_tag_t tag`
- `vtss_vid_t vid`
- `vtss_tagprio_t pcp`
- `vtss_dei_t dei`

7.86.1 Detailed Description

Mirror configuration.

Definition at line 1681 of file vtss_l2_api.h.

7.86.2 Field Documentation

7.86.2.1 port_no

`vtss_port_no_t` `vtss_mirror_conf_t::port_no`

Mirror port or VTSS_PORT_NO_NONE

Definition at line 1683 of file vtss_l2_api.h.

7.86.2.2 fwd_enable

`BOOL` `vtss_mirror_conf_t::fwd_enable`

Enable normal traffic forwarding to mirror port

Definition at line 1684 of file vtss_l2_api.h.

7.86.2.3 tag

`vtss_mirror_tag_t` `vtss_mirror_conf_t::tag`

Mirror tag type

Definition at line 1686 of file vtss_l2_api.h.

7.86.2.4 vid

`vtss_vid_t` `vtss_mirror_conf_t::vid`

Mirror tag VID

Definition at line 1687 of file vtss_l2_api.h.

7.86.2.5 pcp

`vtss_tagprio_t vtss_mirror_conf_t::pcp`

Mirror tag PCP

Definition at line 1688 of file `vtss_l2_api.h`.

7.86.2.6 dei

`vtss_dei_t vtss_mirror_conf_t::dei`

Mirror tag DEI

Definition at line 1689 of file `vtss_l2_api.h`.

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

- `vtss_api/include/vtss_l2_api.h`

7.87 `vtss_mtimer_t` Struct Reference

Timer structure.

```
#include <vtss_os_linux.h>
```

Data Fields

- struct timeval `timeout`
- struct timeval `now`

7.87.1 Detailed Description

Timer structure.

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

7.87.2 Field Documentation

7.87.2.1 timeout

```
struct timeval vtss_mtimer_t::timeout
```

Timeout

Definition at line 89 of file vtss_os_linux.h.

7.87.2.2 now

```
struct timeval vtss_mtimer_t::now
```

Time right now

Definition at line 90 of file vtss_os_linux.h.

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

- vtss_api/include/[vtss_os_linux.h](#)

7.88 vtss_npi_conf_t Struct Reference

NPI configuration.

```
#include <vtss_packet_api.h>
```

Data Fields

- [BOOL enable](#)
- [vtss_port_no_t port_no](#)

7.88.1 Detailed Description

NPI configuration.

Definition at line 48 of file vtss_packet_api.h.

7.88.2 Field Documentation

7.88.2.1 enable

`BOOL vtss_npi_conf_t::enable`

Enable NPI port

Definition at line 49 of file `vtss_packet_api.h`.

7.88.2.2 port_no

`vtss_port_no_t vtss_npi_conf_t::port_no`

Port to use as NPI - if configurable

Definition at line 50 of file `vtss_packet_api.h`.

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

- `vtss_api/include/vtss_packet_api.h`

7.89 `vtss_os_timestamp_t` Struct Reference

```
#include <vtss_misc_api.h>
```

Data Fields

- `unsigned int hw_cnt`

7.89.1 Detailed Description

`VTSS_OS_TIMESTAMP_TYPE` `VTSS_OS_TIMESTAMP()` These two provides a mean to have the API timestamp events for use by the application. It is up to the platform specific code to implement the actual functions to obtain the timestamp. The implementation *must* be callable from interrupt context, so no implicit waits or sleeps are allowed.

Definition at line 1072 of file `vtss_misc_api.h`.

7.89.2 Field Documentation

7.89.2.1 hw_cnt

```
unsigned int vtss_os_timestamp_t::hw_cnt
```

hardware counter

Definition at line 1073 of file vtss_misc_api.h.

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

- [vtss_api/include/vtss_misc_api.h](#)

7.90 vtss_packet_dma_conf_t Struct Reference

```
#include <vtss_packet_api.h>
```

Data Fields

- [BOOL dma_enable \[VTSS_QUEUE_ARRAY_SIZE\]](#)

7.90.1 Detailed Description

The packet DMA configuration.

Definition at line 2124 of file vtss_packet_api.h.

7.90.2 Field Documentation

7.90.2.1 dma_enable

```
BOOL vtss_packet_dma_conf_t::dma_enable[VTSS_QUEUE_ARRAY_SIZE]
```

Enable the given queues for DMA

Definition at line 2125 of file vtss_packet_api.h.

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

- [vtss_api/include/vtss_packet_api.h](#)

7.91 vtss_packet_frame_info_t Struct Reference

Information about frame.

```
#include <vtss_packet_api.h>
```

Data Fields

- `vtss_port_no_t port_no`
- `vtss_glag_no_t glag_no`
- `vtss_vid_t vid`
- `vtss_port_no_t port_tx`
- `BOOL aggr_rx_disable`
- `BOOL aggr_tx_disable`

7.91.1 Detailed Description

Information about frame.

Definition at line 346 of file vtss_packet_api.h.

7.91.2 Field Documentation

7.91.2.1 port_no

```
vtss_port_no_t vtss_packet_frame_info_t::port_no
```

Ingress port (or VTSS_PORT_NO_NONE)

Definition at line 348 of file vtss_packet_api.h.

7.91.2.2 glag_no

```
vtss_glag_no_t vtss_packet_frame_info_t::glag_no
```

Ingress GLAG (or VTSS_GLAG_NO_NONE)

Definition at line 350 of file vtss_packet_api.h.

7.91.2.3 vid

```
vtss_vid_t vtss_packet_frame_info_t::vid
```

Egress VID (or VTSS_VID_NULL)

Definition at line 352 of file vtss_packet_api.h.

7.91.2.4 port_tx

```
vtss_port_no_t vtss_packet_frame_info_t::port_tx
```

Egress port (or VTSS_PORT_NO_NONE)

Definition at line 353 of file vtss_packet_api.h.

7.91.2.5 aggr_rx_disable

```
BOOL vtss_packet_frame_info_t::aggr_rx_disable
```

Disable aggregation Rx filtering

Definition at line 354 of file vtss_packet_api.h.

7.91.2.6 aggr_tx_disable

```
BOOL vtss_packet_frame_info_t::aggr_tx_disable
```

Disable aggregation Tx filtering

Definition at line 355 of file vtss_packet_api.h.

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

- vtss_api/include/[vtss_packet_api.h](#)

7.92 vtss_packet_port_filter_t Struct Reference

Packet information for each port.

```
#include <vtss_packet_api.h>
```

Data Fields

- `vtss_packet_filter_t filter`
- `vtss_etype_t tpid`

7.92.1 Detailed Description

Packet information for each port.

Definition at line 410 of file `vtss_packet_api.h`.

7.92.2 Field Documentation

7.92.2.1 filter

`vtss_packet_filter_t vtss_packet_port_filter_t::filter`

Packet filtering

Definition at line 411 of file `vtss_packet_api.h`.

7.92.2.2 tpid

`vtss_etype_t vtss_packet_port_filter_t::tpid`

Tag Ethernet Type

Definition at line 412 of file `vtss_packet_api.h`.

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

- `vtss_api/include/vtss_packet_api.h`

7.93 `vtss_packet_port_info_t` Struct Reference

Port info structure.

```
#include <vtss_packet_api.h>
```

Data Fields

- `vtss_port_no_t port_no`
- `vtss_glag_no_t glag_no`
- `vtss_vid_t vid`
- `BOOL aggr_rx_disable`
- `BOOL aggr_tx_disable`

7.93.1 Detailed Description

Port info structure.

Definition at line 390 of file `vtss_packet_api.h`.

7.93.2 Field Documentation

7.93.2.1 port_no

`vtss_port_no_t vtss_packet_port_info_t::port_no`

Ingress port (or `VTSS_PORT_NO_NONE`)

Definition at line 391 of file `vtss_packet_api.h`.

7.93.2.2 glag_no

`vtss_glag_no_t vtss_packet_port_info_t::glag_no`

Ingress GLAG (or `VTSS_GLAG_NO_NONE`)

Definition at line 393 of file `vtss_packet_api.h`.

7.93.2.3 vid

`vtss_vid_t vtss_packet_port_info_t::vid`

Egress VID (or `VTSS_VID_NULL`)

Definition at line 395 of file `vtss_packet_api.h`.

7.93.2.4 aggr_rx_disable

`BOOL vtss_packet_port_info_t::aggr_rx_disable`

Disable aggregation Rx filtering

Definition at line 396 of file vtss_packet_api.h.

7.93.2.5 aggr_tx_disable

`BOOL vtss_packet_port_info_t::aggr_tx_disable`

Disable aggregation Tx filtering

Definition at line 397 of file vtss_packet_api.h.

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

- vtss_api/include/[vtss_packet_api.h](#)

7.94 vtss_packet_rx_conf_t Struct Reference

CPU Rx configuration.

```
#include <vtss_packet_api.h>
```

Data Fields

- [vtss_packet_rx_queue_conf_t queue](#) [VTSS_PACKET_RX_QUEUE_CNT]
- [vtss_packet_rx_reg_t reg](#)
- [vtss_packet_rx_queue_map_t map](#)
- [vtss_packet_rx_grp_t grp_map](#) [VTSS_PACKET_RX_QUEUE_CNT]

7.94.1 Detailed Description

CPU Rx configuration.

Definition at line 121 of file vtss_packet_api.h.

7.94.2 Field Documentation

7.94.2.1 queue

```
vtss_packet_rx_queue_conf_t vtss_packet_rx_conf_t::queue[VTSS_PACKET_RX_QUEUE_CNT]
```

Queue configuration

Definition at line 122 of file vtss_packet_api.h.

7.94.2.2 reg

```
vtss_packet_rx_reg_t vtss_packet_rx_conf_t::reg
```

Packet registration

Definition at line 123 of file vtss_packet_api.h.

7.94.2.3 map

```
vtss_packet_rx_queue_map_t vtss_packet_rx_conf_t::map
```

Queue mapping

Definition at line 124 of file vtss_packet_api.h.

7.94.2.4 grp_map

```
vtss_packet_rx_grp_t vtss_packet_rx_conf_t::grp_map[VTSS_PACKET_RX_QUEUE_CNT]
```

Queue to extraction group map

Definition at line 126 of file vtss_packet_api.h.

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

- vtss_api/include/vtss_packet_api.h

7.95 vtss_packet_rx_header_t Struct Reference

System frame header describing received frame.

```
#include <vtss_packet_api.h>
```

Data Fields

- `u32 length`
- `vtss_port_no_t port_no`
- `u32 queue_mask`
- `BOOL learn`
- `BOOL arrived_tagged`
- `vtss_tci_t tag`
- `vtss_vstax_rx_header_t vstax`

7.95.1 Detailed Description

System frame header describing received frame.

Definition at line 216 of file `vtss_packet_api.h`.

7.95.2 Field Documentation

7.95.2.1 `length`

`u32 vtss_packet_rx_header_t::length`

Frame length excluding CRC

Definition at line 218 of file `vtss_packet_api.h`.

7.95.2.2 `port_no`

`vtss_port_no_t vtss_packet_rx_header_t::port_no`

Ingress port number

Definition at line 219 of file `vtss_packet_api.h`.

7.95.2.3 `queue_mask`

`u32 vtss_packet_rx_header_t::queue_mask`

Bitmask of queues where received on

Definition at line 220 of file `vtss_packet_api.h`.

7.95.2.4 learn

`BOOL vtss_packet_rx_header_t::learn`

TRUE if learn frame

Definition at line 221 of file vtss_packet_api.h.

7.95.2.5 arrived_tagged

`BOOL vtss_packet_rx_header_t::arrived_tagged`

TRUE if frame was tagged

Definition at line 222 of file vtss_packet_api.h.

7.95.2.6 tag

`vtss_tci_t vtss_packet_rx_header_t::tag`

VLAN tag from frame or port setup

Definition at line 223 of file vtss_packet_api.h.

7.95.2.7 vstax

`vtss_vstax_rx_header_t vtss_packet_rx_header_t::vstax`

VStaX header

Definition at line 225 of file vtss_packet_api.h.

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

- vtss_api/include/[vtss_packet_api.h](#)

7.96 vtss_packet_rx_info_t Struct Reference

Decoded extraction header properties.

```
#include <vtss_packet_api.h>
```

Data Fields

- `u32 hints`
- `u32 length`
- `vtss_port_no_t port_no`
- `vtss_glag_no_t glag_no`
- `vtss_tag_type_t tag_type`
- `vtss_vlan_tag_t tag`
- `vtss_vlan_tag_t stripped_tag`
- `u32 xtr_qu_mask`
- `vtss_prio_t cos`
- `BOOL acl_hit`
- `u32 acl_idx`
- `VTSS_OS_TIMESTAMP_TYPE sw_tstamp`
- `u32 tstamp_id`
- `BOOL tstamp_id_decoded`
- `u32 hw_tstamp`
- `BOOL hw_tstamp_decoded`
- `vtss_sflow_type_t sflow_type`
- `vtss_port_no_t sflow_port_no`
- `u64 oam_info`
- `BOOL oam_info_decoded`
- `vtss_isdx_t isdx`
- `vtss_vstax_rx_header_t vstax`

7.96.1 Detailed Description

Decoded extraction header properties.

This structure gets populated with a call to `vtss_packet_rx_hdr_decode()`.

Many decoded parameters have two fields in the structure: One indicating the value of the parameter (e.g. `tstamp_id`), and another indicating if the parameter is actually decoded or not (e.g. `tstamp_id_decoded`).

The reason for having an XXX_decoded boolean for every parameter is that the information held in the extraction header is very different on the various Vitesse chips, and even on the same chip type, the information may be overloaded, depending on the incoming frame type.

Most parameters don't have a decoded field, and in that case, they are always decoded.

Definition at line 1000 of file `vtss_packet_api.h`.

7.96.2 Field Documentation

7.96.2.1 hints

`u32 vtss_packet_rx_info_t::hints`

The `hints` member is useful for applications that wish to perform some kind of ingress filtering on received frames. Please refer to `vtss_packet_rx_hints_t` for a full description. Each of the enumerations can be bitwise ORed into the `hints` member.

Validity:

Luton26: Y
Jaguar1: Y
Serval : Y
Jaguar2: Y
Serval2: Y
ServalT: Y

Definition at line 1017 of file `vtss_packet_api.h`.

7.96.2.2 length

`u32 vtss_packet_rx_info_t::length`

The length of the frame from DMAC to end-of-frame excluding FCS.

This is a copy of `vtss_packet_rx_meta_t::length`.

Validity:

Luton26: Y
Jaguar1: Y
Serval : Y
Jaguar2: Y
Serval2: Y
ServalT: Y

Definition at line 1034 of file `vtss_packet_api.h`.

7.96.2.3 port_no

`vtss_port_no_t vtss_packet_rx_info_t::port_no`

The logical source port on which the frame was received. In a few cases, this may be `VTSS_PORT_NO_NONE`, if the physical source port was not part of the port map (e.g. in JR-48, where sFlow frames were Tx sampled and received on an interconnect port).

Validity:

Luton26: Y
Jaguar1: Y
Serval : Y
Jaguar2: Y
Serval2: Y
ServalT: Y

Definition at line 1052 of file `vtss_packet_api.h`.

7.96.2.4 glag_no

`vtss_glag_no_t vtss_packet_rx_info_t::glag_no`

The Global Link Aggregation Group this frame was received on. VTSS_GLAG_NO_NONE if not received on a GLAG (also on non-supporting architectures). If received on a GLAG, `port_no` will contain the first member port in the GLAG.

If received on a stack port, `glag_no` will always be VTSS_GLAG_NO_NONE, but if the sender supports it in S/W, the stack header can be filled with a glag_no before it is transmitted. To obtain this glag_no on the receiving side, you can find it in vstax member's glag_no member.

Validity:

Luton26:	N
Jaguar1:	Y
Serval :	N
Jaguar2:	Y
Serval2:	N
ServalT:	N

Definition at line 1076 of file vtss_packet_api.h.

7.96.2.5 tag_type

`vtss_tag_type_t vtss_packet_rx_info_t::tag_type`

The tag type (802.1Q TPID) the frame was received with, if any.

Not all architectures support all four enumerations of the tag type.

On all architectures, the tag stays in the frame until received by S/W.

In general, it works like this: If a frame is received on a VLAN unaware port, `tag_type` will always be set to VTSS_TAG_TYPE_UNTAGGED, whether it contains a tag or not. The classified VLAN (`tag`'s vid member) will always be the port VID.

If a frame is received on a C-port, then only frames received with a C-tag are marked as VTSS_TAG_TYPE_C_TAGGED. S- and S-custom-tagged frames will be marked as VTSS_TAG_TYPE_UNTAGGED, but notice that the frame *may* be classified according to that tag, anyway (architecture dependent). The `hints` `vlan_tag_mismatch` member will be set to TRUE to indicate such a condition.

The same goes for frames received on an S-port or S-custom-port with a "foreign" VLAN tag. The frame may on some architectures be classified to the VID in the tag, but on others be classified according to the PVID.

If the FDMA driver is being used for extracting frames, it can be configured to drop frames received with a wrong tag. It can also be configured to strip tags if received with a tag according to its port setup.

Validity:

Luton26:	Y
Jaguar1:	Y
Serval :	Y
Jaguar2:	Y
Serval2:	Y
ServalT:	Y

Definition at line 1118 of file vtss_packet_api.h.

7.96.2.6 tag

`vtss_vlan_tag_t vtss_packet_rx_info_t::tag`

VLAN classification information.

Contains the classified VLAN information, as opposed to [stripped_tag](#), which contains the actual VLAN tag as was in the frame.

Only the .pcp, .dei, and .vid members are used. Notice that this is not necessarily the classification coming from a VLAN tag; it may come from normal port classification.

Validity:

Luton26: Y
Jaguar1: Y
Serval : Y
Jaguar2: Y
Serval2: Y
ServalT: Y

Definition at line 1141 of file vtss_packet_api.h.

7.96.2.7 stripped_tag

`vtss_vlan_tag_t vtss_packet_rx_info_t::stripped_tag`

VLAN stripped tag information.

Opposed to [tag](#), [stripped_tag](#) contains the VLAN information coming directly from the frame. The classified and frame VLAN information may differ due to ACL/VCAP rules that causes the frame to get classified based on other properties than a possible VLAN tag embedded in the frame.

Whenever the .tpid member is non-zero, the tag was stripped from the frame.

NOTICE: Only platforms that use the FDMA driver will have the ability to strip tags, and this will only happen when the FDMA driver is configured for it (this is the default, though).

Validity (FDMA only) :

Luton26: Y
Jaguar1: Y
Serval : Y
Jaguar2: Y
Serval2: Y
ServalT: Y

Definition at line 1169 of file vtss_packet_api.h.

7.96.2.8 xtr_qu_mask

`u32 vtss_packet_rx_info_t::xtr_qu_mask`

CPU extraction queue mask (one bit per CPU extraction queue). Each bit implies the frame was subject to CPU forwarding to the specific queue. The actual queue the frame was received on is given by the most significant bit set in the mask.

Validity:

Luton26:	Y
Jaguar1:	Y (but in some cases, it is constructed rather than showing the true story (constructed)
Serval :	Y
Jaguar2:	Y
Serval2:	Y
ServalT:	Y

Definition at line 1188 of file vtss_packet_api.h.

7.96.2.9 cos

`vtss_prio_t vtss_packet_rx_info_t::cos`

The frame's classified QoS class.

Validity:

Luton26:	Y
Jaguar1:	Y
Serval :	Y
Jaguar2:	Y
Serval2:	Y
ServalT:	Y

Definition at line 1203 of file vtss_packet_api.h.

7.96.2.10 acl_hit

`BOOL vtss_packet_rx_info_t::acl_hit`

Set if frame has hit a rule in IS2, which copies the frame to the CPU (IS2 actions CPU_COPY_ENA or HIT_ME_↔ ONCE). `acl_idx` may contain the IS2 entry number. For Serval, the `acl_idx` contains the combined ACL_ID action of the rules hit in IS2.

Validity:

Luton26:	Y
Jaguar1:	Y
Serval :	Y
Jaguar2:	Y
Serval2:	Y
ServalT:	Y

Definition at line 1222 of file vtss_packet_api.h.

7.96.2.11 acl_idx

`u32 vtss_packet_rx_info_t::acl_idx`

If `acl_hit` is set, this value is the entry number of the rule hit in IS2. If both IS2 lookups hit a rule which copy the frame to the CPU, then the second lookup's entry number is used.

For Serval, this is the combined ACL_ID action coming out of the two IS2 look-ups.

Validity:

Luton26:	Y
Jaguar1:	Y
Serval :	Y
Jaguar2:	N
Serval2:	N
ServalT:	N

Definition at line 1242 of file vtss_packet_api.h.

7.96.2.12 sw_tstamp

`VTSS_OS_TIMESTAMP_TYPE vtss_packet_rx_info_t::sw_tstamp`

Software timestamp of packet.

This is a copy of the `vtss_packet_rx_meta_t::sw_tstamp` field.

Validity:

Luton26:	Y
Jaguar1:	Y
Serval :	Y
Jaguar2:	Y
Serval2:	Y
ServalT:	Y

Definition at line 1259 of file vtss_packet_api.h.

7.96.2.13 tstamp_id

`u32 vtss_packet_rx_info_t::tstamp_id`

Two-step PTP timestamp identifier (6 bits).

On Luton26, this field identifies an Rx timestamp and potential Tx timestamps (if the PTP frame was forwarded to other ports). Notice that `tstamp_id_decoded` will be TRUE for all frames that have hit in IS2 rule. This means that the application must make additional checks that this indeed is a PTP frame before relying on `tstamp_id`.

On Serval, this field identifies Tx timestamps. Rx timestamps are embedded in the extraction header.

Validity:

Luton26:	Y
Jaguar1:	N
Serval :	Y
Jaguar2:	N
Serval2:	N
ServalT:	N

Definition at line 1284 of file vtss_packet_api.h.

7.96.2.14 tstamp_id_decoded

`BOOL vtss_packet_rx_info_t::tstamp_id_decoded`

TRUE when `tstamp_id` contains valid information, FALSE otherwise.

Definition at line 1288 of file `vtss_packet_api.h`.

7.96.2.15 hw_tstamp

`u32 vtss_packet_rx_info_t::hw_tstamp`

The frame's ingress timestamp.

Jaguar1: Frames getting copied to the CPU for SFlow reasons can never have a valid `hw_tstamp` (such frames will be indicated through `hw_tstamp_decoded == FALSE`). Note that the `hw_tstamp_decoded` will indicate TRUE for all other frames, but this is only reliable if the following register are set-up: ANA_AC:PS_COMMON:MISC_CTRL.OAM_RX_FRM_EXTEND_ENA == 0 ANA_AC:PS_COMMON:MISC_CTRL.OAM_RX_TSTAMP_IN_FCS_ENA == 1 ASM:CFG:ETH_CFG.ETH_PRE_MODE == 1 DEV1G/DEV25G:DEV_CFG_STATUS:DEV_PTP_CFG.PTP_ENA == 1 DEV10G:DEV_CFG_STATUS:DEV_PTP_CFG.PTP_CFG_ENA == 1 DEVCPU_GCB:PTP_CFG:PTP_MIS_C_CFG.PTP_ENA == 1

Serval: Two-step PTP frames have the 32-bit Rx timestamp saved in this field.

Validity:

Luton26:	N
Jaguar1:	Y
Serval :	Y
Jaguar2:	Y
Serval2:	Y
ServalT:	Y

Definition at line 1318 of file `vtss_packet_api.h`.

7.96.2.16 hw_tstamp_decoded

`BOOL vtss_packet_rx_info_t::hw_tstamp_decoded`

TRUE when `hw_tstamp` contains valid information, FALSE otherwise.

Definition at line 1322 of file `vtss_packet_api.h`.

7.96.2.17 sflow_type

```
vtss_sf_low_type_t vtss_packet_rx_info_t::sflow_type
```

sFlow type. Indicates if this is copied to the CPU due to Rx or Tx SFlow, or if it's not due to sFlow in the first place (VTSS_SFLOW_TYPE_NONE).

Only VTSS_SFLOW_TYPE_NONE, VTSS_SFLOW_TYPE_RX, and VTSS_SFLOW_TYPE_TX are possible.

Jaguar1 + Jaguar2: Note: [sflow_type](#)'s RX and TX enumerations are only reliable if ANA_AC:PS_COMMON:PS_COMMON_CFG.SFLOW_SMPL_ID_IN_STAMP_ENA is set to 1. However, [sflow_type](#) == VTSS_SFLOW_TYPE_NONE is always reliable.

Validity:

Luton26:	Y
Jaguar1:	Y
Serval :	Y
Jaguar2:	Y
Serval2:	Y
Servalt:	Y

Definition at line 1346 of file vtss_packet_api.h.

7.96.2.18 sflow_port_no

```
vtss_port_no_t vtss_packet_rx_info_t::sflow_port_no
```

sFlow port. Indicates the logical sFlow Rx or Tx port number that caused this frame to be copied to the CPU. Only valid if [sflow_type](#) != VTSS_SFLOW_TYPE_NONE.

Jaguar1 + Jaguar2: Note: This is only reliable if ANA_AC:PS_COMMON:PS_COMMON_CFG.SFLOW_SMPL_ID_IN_STAMP_ENA is set to 1. That bit must be set if Tx sFlow are enabled. If only using Rx sFlows, that bit can be cleared, and you may use the [port_no](#) member to figure out which port caused this frame.

Validity:

Luton26:	Y
Jaguar1:	Y
Serval :	Y
Jaguar2:	Y
Serval2:	Y
Servalt:	Y

Definition at line 1368 of file vtss_packet_api.h.

7.96.2.19 oam_info

```
u64 vtss_packet_rx_info_t::oam_info
```

Various un-decodable OAM info. Decoding of the OAM info field from the extraction header requires accompanying frame info, and is therefore saved as an opaque type, letting it be up to the application to decode it based on description in the datasheet.

Serval: This corresponds to the contents of the REW_VAL field in the extraction header. oam_info_decoded = TRUE when REW_OP[2:0] == 4. Only the 32 lsbits of [oam_info](#) are used.

Validity:

Luton26:	N
Jaguar1:	N
Serval :	Y
Jaguar2:	N
Serval2:	N
ServalT:	N

Definition at line 1393 of file vtss_packet_api.h.

7.96.2.20 oam_info_decoded

```
BOOL vtss_packet_rx_info_t::oam_info_decoded
```

TRUE when [oam_info](#) contains valid information, FALSE otherwise.

Definition at line 1397 of file vtss_packet_api.h.

7.96.2.21 isdx

```
vtss_isdx_t vtss_packet_rx_info_t::isdx
```

The N-bit ISDX from IS1 classification, or VTSS_ISDX_NONE.

Validity:

Luton26:	N
Jaguar1:	N
Serval :	Y
Jaguar2:	N
Serval2:	N
ServalT:	N

Definition at line 1412 of file vtss_packet_api.h.

7.96.2.22 vstax

`vtss_vstax_rx_header_t vtss_packet_rx_info_t::vstax`

The frame's decoded VStaX header. `vtss_vstax_rx_header_t::valid` indicates whether the frame was received with a VStaX header.

Validity:

Luton26: N
Jaguar1: Y
Serval : N
Jaguar2: Y
Serval2: N
ServalT: N

Definition at line 1430 of file vtss_packet_api.h.

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

- `vtss_api/include/vtss_packet_api.h`

7.97 vtss_packet_rx_meta_t Struct Reference

Input structure to `vtss_packet_rx_hdr_decode()`.

```
#include <vtss_packet_api.h>
```

Data Fields

- `BOOL no_wait`
- `vtss_chip_no_t chip_no`
- `vtss_packet_rx_queue_t xtr_qu`
- `vtss_etype_t etype`
- `u32 fcs`
- `VTSS_OS_TIMESTAMP_TYPE sw_tstamp`
- `u32 length`

7.97.1 Detailed Description

Input structure to `vtss_packet_rx_hdr_decode()`.

In order to be able to decode the side-band information coming with a frame when extracting through an external CPU with extraction headers enabled, a number of inputs are required.

This structure provides this meta data.

For future compatibility, `memset()` this structure to 0 prior to filling it in.

Definition at line 727 of file vtss_packet_api.h.

7.97.2 Field Documentation

7.97.2.1 no_wait

`BOOL vtss_packet_rx_meta_t::no_wait`

This function may provide trace output as it decodes the header. Since it may be called from various contexts, hereunder interrupt (DSR) if e.g. used by the FDMA driver, it's crucial that it calls the correct trace print function, that is, one that can print without making waiting points. If `no_wait` is TRUE, the trace output will be directed to VTSS_TRACE_GROUP_FDMA_IRQ, which is assumed to be able to print directly to the console without waiting. If `no_wait` is FALSE, the trace output will be directed to VTSS_TRACE_GROUP_PACKET, which is assumed to print from thread context.

Required to be set?

```
Luton26: Y
Jaguar1: Y
Serval: Y
Jaguar2: Y
Serval2: Y
ServalT: Y
```

Definition at line 755 of file vtss_packet_api.h.

7.97.2.2 chip_no

`vtss_chip_no_t vtss_packet_rx_meta_t::chip_no`

Chip number on which this frame was extracted. It is not possible to deduct from the binary extraction header which device the frame was extracted. In order to be able to provide a logical source port in the decoded extraction properties, a chip number is required for targets made up of multiple physical chips.

Required to be set?

```
Luton26: N (assumed to be 0)
Jaguar1: Y
Serval: N (assumed to be 0)
Jaguar2: N (assumed to be 0)
Serval2: N (assumed to be 0)
ServalT: N (assumed to be 0)
```

Definition at line 777 of file vtss_packet_api.h.

7.97.2.3 xtr_qu

`vtss_packet_rx_queue_t vtss_packet_rx_meta_t::xtr_qu`

Rx queue number from which this frame was really extracted. This is only needed on particular architectures, where this info is not part of the extraction header.

Jaguar1: Frames received in super priority queues (queue 8 and 9), do not contain extraction headers. Only the FDMA driver and external CPUs using register-based readings are able to extract super priority frames. Such applications will have to construct a synthetic IFH in order to get it decoded with this function. Also, in cases where a frame hits a non-default IS2 rule, the extraction queue mask in the IFH is overloaded with an ACL index. Therefore, in order to be able to construct an extraction queue mask in the decoded extraction header, `xtr_qu` should be filled in.

Required to be set?

```
Luton26: N
Jaguar1: Y
Serval: N
Jaguar2: N
Serval2: N
ServalT: N
```

Definition at line 808 of file vtss_packet_api.h.

7.97.2.4 etype

`vtss_etype_t vtss_packet_rx_meta_t::etype`

The Ethernet type of the received frame.

This is needed in order to be able to decode `vtss_packet_rx_info_t::tag_type` correctly.

Required to be set?

```
Luton26: Y
Jaguar1: Y
Serval: Y
Jaguar2: Y
Serval2: Y
ServalT: Y
```

Definition at line 827 of file vtss_packet_api.h.

7.97.2.5 fcs

`u32 vtss_packet_rx_meta_t::fcs`

Frame checksum.

On some architectures, the frame's FCS/CRC may be used to hold additional side-band information about the frame itself.

To be able to decode this information, the caller must therefore find the end of the frame and extract the 32-bit FCS and place it here.

Required to be set?

```
Luton26: N
Jaguar1: Y (sflow-info or timestamp)
Serval: N
Jaguar2: Y (sfflow-info)
Serval2: Y (sfflow-info)
ServalT: Y (sfflow-info)
```

Definition at line 851 of file `vtss_packet_api.h`.

7.97.2.6 sw_tstamp

`VTSS_OS_TIMESTAMP_TYPE vtss_packet_rx_meta_t::sw_tstamp`

Software timestamp of packet.

This may be used by an external CPU to S/W-wise timestamp the packet. If the FDMA driver is being used to extract frames, it will take care of filling this field in.

The field will be copied directly to [vtss_packet_rx_info_t::sw_tstamp](#).

Required to be set?

```
Luton26: N
Jaguar1: N
Serval: N
Jaguar2: N
Serval2: N
ServalT: N
```

Definition at line 873 of file `vtss_packet_api.h`.

7.97.2.7 length

`u32 vtss_packet_rx_meta_t::length`

The length of the frame from DMAC to end-of-frame excluding FCS.

The extraction header normally doesn't include the frame length, so in order to provide a structure to the application that contains all meta data about any given frame, the application may fill in this member, which will be copied exactly as is into `vtss_packet_rx_info_t::length`.

If the FDMA driver is used to extract frames, it will take care of filling this field in.

Required to be set?

Luton26: N
Jaguar1: N
Serval: N
Jaguar2: N
Serval2: N
ServalT: N

Definition at line 897 of file vtss_packet_api.h.

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

- `vtss_api/include/vtss_packet_api.h`

7.98 vtss_packet_rx_port_conf_t Struct Reference

Packet registration per port.

```
#include <types.h>
```

Data Fields

- `vtss_packet_reg_type_t bpdu_reg` [16]
- `vtss_packet_reg_type_t garp_reg` [16]

7.98.1 Detailed Description

Packet registration per port.

Definition at line 764 of file types.h.

7.98.2 Field Documentation

7.98.2.1 bpdu_reg

`vtss_packet_reg_type_t vtss_packet_rx_port_conf_t::bpdu_reg[16]`

BPDU range: 01-80-C2-00-00-0X

Definition at line 770 of file types.h.

7.98.2.2 garp_reg

`vtss_packet_reg_type_t vtss_packet_rx_port_conf_t::garp_reg[16]`

GARP range: 01-80-C2-00-00-2X

Definition at line 771 of file types.h.

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

- `vtss_api/include/vtss/api/types.h`

7.99 vtss_packet_rx_queue_conf_t Struct Reference

CPU Rx queue configuration.

```
#include <vtss_packet_api.h>
```

Data Fields

- `vtss_packet_rx_queue_size_t size`
- `vtss_packet_rx_queue_npi_conf_t npi`

7.99.1 Detailed Description

CPU Rx queue configuration.

Definition at line 85 of file vtss_packet_api.h.

7.99.2 Field Documentation

7.99.2.1 size

`vtss_packet_rx_queue_size_t vtss_packet_rx_queue_conf_t::size`

Queue size

Definition at line 86 of file vtss_packet_api.h.

7.99.2.2 npi

`vtss_packet_rx_queue_npi_conf_t vtss_packet_rx_queue_conf_t::npi`

NPI configuration

Definition at line 88 of file vtss_packet_api.h.

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

- `vtss_api/include/vtss_packet_api.h`

7.100 vtss_packet_rx_queue_map_t Struct Reference

CPU Rx queue map.

```
#include <vtss_packet_api.h>
```

Data Fields

- `vtss_packet_rx_queue_t bpdu_queue`
- `vtss_packet_rx_queue_t garp_queue`
- `vtss_packet_rx_queue_t learn_queue`
- `vtss_packet_rx_queue_t igmp_queue`
- `vtss_packet_rx_queue_t ipmc_ctrl_queue`
- `vtss_packet_rx_queue_t mac_vid_queue`
- `vtss_packet_rx_queue_t stack_queue`
- `vtss_packet_rx_queue_t sflow_queue`
- `vtss_packet_rx_queue_t lrn_all_queue`
- `vtss_packet_rx_queue_t l3_uc_queue`
- `vtss_packet_rx_queue_t l3_other_queue`

7.100.1 Detailed Description

CPU Rx queue map.

Definition at line 103 of file vtss_packet_api.h.

7.100.2 Field Documentation

7.100.2.1 bpdu_queue

`vtss_packet_rx_queue_t vtss_packet_rx_queue_map_t::bpdu_queue`

BPDUs

Definition at line 105 of file `vtss_packet_api.h`.

7.100.2.2 garp_queue

`vtss_packet_rx_queue_t vtss_packet_rx_queue_map_t::garp_queue`

GARP frames

Definition at line 106 of file `vtss_packet_api.h`.

7.100.2.3 learn_queue

`vtss_packet_rx_queue_t vtss_packet_rx_queue_map_t::learn_queue`

Learn frames

Definition at line 107 of file `vtss_packet_api.h`.

7.100.2.4 igmp_queue

`vtss_packet_rx_queue_t vtss_packet_rx_queue_map_t::igmp_queue`

IGMP/MLD frames

Definition at line 108 of file `vtss_packet_api.h`.

7.100.2.5 ipmc_ctrl_queue

`vtss_packet_rx_queue_t vtss_packet_rx_queue_map_t::ipmc_ctrl_queue`

IP multicast control frames

Definition at line 109 of file `vtss_packet_api.h`.

7.100.2.6 mac_vid_queue

`vtss_packet_rx_queue_t` `vtss_packet_rx_queue_map_t::mac_vid_queue`

MAC address table

Definition at line 110 of file vtss_packet_api.h.

7.100.2.7 stack_queue

`vtss_packet_rx_queue_t` `vtss_packet_rx_queue_map_t::stack_queue`

CPU-generated VStaX traffic

Definition at line 111 of file vtss_packet_api.h.

7.100.2.8 sflow_queue

`vtss_packet_rx_queue_t` `vtss_packet_rx_queue_map_t::sflow_queue`

SFlow-marked frames

Definition at line 112 of file vtss_packet_api.h.

7.100.2.9 lrn_all_queue

`vtss_packet_rx_queue_t` `vtss_packet_rx_queue_map_t::lrn_all_queue`

Learn-all queue - JR-48 and JR-Stacking only

Definition at line 113 of file vtss_packet_api.h.

7.100.2.10 l3_uc_queue

`vtss_packet_rx_queue_t` `vtss_packet_rx_queue_map_t::l3_uc_queue`

L3 routing unicast queue

Definition at line 115 of file vtss_packet_api.h.

7.100.2.11 l3_other_queue

`vtss_packet_rx_queue_t vtss_packet_rx_queue_map_t::l3_other_queue`

L3 routing other frames queue

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

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

- `vtss_api/include/vtss_packet_api.h`

7.101 `vtss_packet_rx_queue_npi_conf_t` Struct Reference

CPU Rx queue NPI configuration.

```
#include <vtss_packet_api.h>
```

Data Fields

- `BOOL enable`

7.101.1 Detailed Description

CPU Rx queue NPI configuration.

Definition at line 75 of file `vtss_packet_api.h`.

7.101.2 Field Documentation

7.101.2.1 enable

`BOOL vtss_packet_rx_queue_npi_conf_t::enable`

Enable redirect of frames to NPI port

Definition at line 76 of file `vtss_packet_api.h`.

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

- `vtss_api/include/vtss_packet_api.h`

7.102 vtss_packet_rx_reg_t Struct Reference

CPU Rx packet registration.

```
#include <vtss_packet_api.h>
```

Data Fields

- `BOOL bpdu_cpu_only`
- `BOOL garp_cpu_only [16]`
- `BOOL ipmc_ctrl_cpu_copy`
- `BOOL igmp_cpu_only`
- `BOOL mld_cpu_only`

7.102.1 Detailed Description

CPU Rx packet registration.

Definition at line 93 of file vtss_packet_api.h.

7.102.2 Field Documentation

7.102.2.1 bpdu_cpu_only

```
BOOL vtss_packet_rx_reg_t::bpdu_cpu_only
```

Redirect BPDUs (DMAC 01-80-C2-00-00-0X)

Definition at line 95 of file vtss_packet_api.h.

7.102.2.2 garp_cpu_only

```
BOOL vtss_packet_rx_reg_t::garp_cpu_only[16]
```

Redirect GARP (DMAC 01-80-C2-00-00-2X)

Definition at line 96 of file vtss_packet_api.h.

7.102.2.3 ipmc_ctrl_cpu_copy

`BOOL vtss_packet_rx_reg_t::ipmc_ctrl_cpu_copy`

Copy IP MC control (DIP 224.0.0.x) to CPU

Definition at line 97 of file vtss_packet_api.h.

7.102.2.4 igmp_cpu_only

`BOOL vtss_packet_rx_reg_t::igmp_cpu_only`

Redirect IGMP frames to the CPU

Definition at line 98 of file vtss_packet_api.h.

7.102.2.5 mld_cpu_only

`BOOL vtss_packet_rx_reg_t::mld_cpu_only`

Redirect MLD frames to the CPU

Definition at line 99 of file vtss_packet_api.h.

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

- `vtss_api/include/vtss_packet_api.h`

7.103 vtss_packet_tx_ifh_t Struct Reference

Compiled Tx Frame Header.

```
#include <vtss_packet_api.h>
```

Data Fields

- `u32 length`
- `u32 ifh [VTSS_PACKET_TX_IFH_MAX/4]`

7.103.1 Detailed Description

Compiled Tx Frame Header.

This is a pre-compiled representation of injection properties similar to those given by `vtss_packet_tx_hdr_encode()`, but wrapped in the following structure. The structure is useful for preallocation, and can be used in combination with `vtss_packet_tx_frame()`.

Definition at line 2072 of file vtss_packet_api.h.

7.103.2 Field Documentation

7.103.2.1 length

```
u32 vtss_packet_tx_ifh_t::length
```

Length of compiled IFH (in bytes)

Definition at line 2073 of file vtss_packet_api.h.

7.103.2.2 ifh

```
u32 vtss_packet_tx_ifh_t::ifh[VTSS_PACKET_TX_IFH_MAX/4]
```

Compiled, binary IFH

Definition at line 2074 of file vtss_packet_api.h.

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

- vtss_api/include/vtss_packet_api.h

7.104 vtss_packet_tx_info_t Struct Reference

Injection Properties.

```
#include <vtss_packet_api.h>
```

Data Fields

- `BOOL switch_frm`
- `u64 dst_port_mask`
- `u32 frm_len`
- `vtss_vlan_tag_t tag`
- `u8 aggr_code`
- `vtss_prio_t cos`
- `vtss_packet_tx_vstax_t tx_vstax_hdr`
- union {
 - u8 bin [VTSS_VSTAX_HDR_SIZE]
 - `vtss_vstax_tx_header_t sym`}
- `vtss_packet_ptp_action_t ptp_action`
- `u8 ptp_id`
- `u32 ptp_timestamp`
- `u32 latch_timestamp`
- `vtss_packet_oam_type_t oam_type`
- `vtss_isdx_t isdx`
- `BOOL isdx_dont_use`
- `vtss_dp_level_t dp`
- `vtss_port_no_t masquerade_port`
- `u32 pdu_offset`

7.104.1 Detailed Description

Injection Properties.

Structural properties used to compose a binary injection header useful for injection into an injection-header-enabled port on the switch.

This structure must be initialized with [vtss_packet_tx_info_init\(\)](#) prior to calling [vtss_packet_tx_hdr_encode\(\)](#).

For each property, it is indicated which platforms the property is valid for. There are two columns, one named 'A' and another named 'F'. 'A' stands for 'API' and indicates whether the [vtss_packet_tx_hdr_encode\(\)](#) function itself utilizes the value. 'F' stands for 'FDMA' and indicates whether the FDMA driver (which also uses this structure, and indirectly also the [vtss_packet_tx_hdr_encode\(\)](#) function) requires this value to be set in its "inject packet API" call.

Definition at line 1462 of file vtss_packet_api.h.

7.104.2 Field Documentation

7.104.2.1 switch_frm

`BOOL vtss_packet_tx_info_t::switch_frm`

If FALSE, the analyzer processing is skipped for this frame, and the destination port set is specified with [dst_port_mask](#). If TRUE, the frame will be passed through the analyzer to find the destination port set. The analyzer looks up the DMAC in the MAC table and forwards based on its findings. The frame may therefore get flooded.

Luton26, Jaguar1, and Serval: If [switch_frm](#) is TRUE, the frame must have inserted a VLAN tag into the frame prior to transmission, to get it classified to the correct VLAN. There is one exception to this, namely when masquerading (see [masquerade_port](#)), where a VLAN tag may or may not be inserted. If not inserted, the masquerade port's PVID is used for classification. In no VLAN tag is inserted by the application/FDMA driver, this structure's [tag](#) member's tpid must be set to 0.

If FALSE, the destination port set must be specified with [dst_port_mask](#).

```
Validity: A F
-----
Luton26: Y Y
Jaguar1: Y Y
Serval : Y Y
Jaguar2: Y Y
Serval2: Y Y
ServalT: Y Y
```

On Luton26 + Serval: If FALSE the rewriter still uses the tag information for rewriting actions.

Definition at line 1493 of file vtss_packet_api.h.

7.104.2.2 dst_port_mask

```
u64 vtss_packet_tx_info_t::dst_port_mask
```

This field provides the logical destination port set onto which to send the frame and may thus be used to multicast the same frame on multiple front ports in one go. The field is only used if [switch_frm](#) is FALSE.

If the frame is going to be transmitted with a VStaX header (tx_vstax_hdr is != VTSS_PACKET_TX_VSTAX_NONE) the dst_port_mask must have exactly one bit set, representing the stack port (A or B) to transmit the frame to. Also, if the frame is subject to periodic transmission through the AFI, exactly one bit must be set.

Jaguar1: In 48-port solutions, only one bit can be set if hitting the secondary device, unless using the FDMA driver v. 3, which will take care of injecting the frame multiple times.

Validity: A F

```
Luton26: Y Y
Jaguar1: Y Y
Serval : Y Y
Jaguar2: Y Y
Serval2: Y Y
ServalT: Y Y
```

Definition at line 1522 of file vtss_packet_api.h.

7.104.2.3 frm_len

```
u32 vtss_packet_tx_info_t::frm_len
```

On some architectures, the frame length must be specified in the injection header. The length - in bytes - is the size of the frame starting from the DMAC up to, but not including, the FCS/CRC.

Validity: A F

```
Luton26: N N
Jaguar1: N N
Serval : N N
Jaguar2: N N
Serval2: N N
ServalT: N N
```

Definition at line 1540 of file vtss_packet_api.h.

7.104.2.4 tag

`vtss_vlan_tag_t vtss_packet_tx_info_t::tag`

VLAN tag information.

Use of this field is architecture specific, and depends on whether `switch_frm` is TRUE or FALSE.

`switch_frm == TRUE`: The frame will get classified according to tag.vid, tag.dei, and tag.pcp.

An application that directly calls `vtss_packet_tx_hdr_encode()` must insert a VLAN tag into the frame with these properties, and the `vtss_packet_tx_hdr_encode()` function will not use `tag` for anything. If the application uses the FDMA, the FDMA driver code will insert the tag into the frame.

When masquerading, the FDMA driver can be controlled not to insert a VLAN tag into the frame by setting tag.vid to 0. See also `masquerade_port`.

`switch_frm == FALSE`: On Serval, egress ES0 lookups occur even when the rewriter is disabled. The result of the lookup may be used in e.g. the VOE to count frames. In order to get the VOE to count in the correct buckets, the tag's pcp member must be set correctly.

If `tag`'s tpid member is non-zero, the FDMA driver inserts a tag into the frame (this must be done by application software if running without the FDMA). In this case, the chip's rewriter will be disabled, so the frame will be sent as is. The inserted tag will use all members of `tag` (i.e. tag.tpid, tag.vid, tag.pcp, and tag.dei).

If `tag`'s tpid member is zero, the FDMA driver will not insert a tag into the frame.

The following applies to all platforms but Jaguar1:

If tpid is zero, rewriting of the frame can now be controlled with `tag`'s vid member: If vid is zero, the chip's rewriter will be disabled, so the frame will be sent as is. If vid is non-zero, the vid will be inserted into the injection header's classified VID field, and the chip's rewriter will be enabled, causing the frame to potentially be rewritten by the chip.

Validity: A F

```
Luton26: N Y
Jaguar1: N Y
Serval : Y Y (A: tag.pcp when switch_frm == FALSE, F: Always).
Jaguar2: N Y
Serval2: N Y
ServalT: N Y
```

Definition at line 1594 of file `vtss_packet_api.h`.

7.104.2.5 aggr_code

`u8 vtss_packet_tx_info_t::aggr_code`

The aggregation code that this frame will use. The 4-bit number maps directly to the corresponding field in the IFH, but is only set if the frame is being switched (the `switch_frm` member is TRUE) and the destination port number is `VTSS_CPU_PM_NUMBER`.

If using the FDMA driver, the application code does not need set it.

Validity: A F

```
Luton26: N N
Jaguar1: N N
Serval : N N
Jaguar2: N N
Serval2: N N
ServalT: N N
```

Definition at line 1616 of file vtss_packet_api.h.

7.104.2.6 cos

```
vtss_prio_t vtss_packet_tx_info_t::cos
```

The QoS class that this frame will be transmitted on. This is a number in range [0; 8].

If you set it to '8' (or VTSS_PRIO_SUPER), the frame will be transmitted with super-priority, i.e. with even higher priority than the highest priority achievable for normal data traffic (if supported on the platform). This is not a valid setting if [switch_frm == TRUE](#).

Luton26 notes: A value of '8' (super priority) will be converted to a value of '7'.

Jaguar1 notes: For switched frames ([switch_frm == TRUE](#)), [cos](#) goes into the VLAN tag's PCP in order to have the switch core classify it to the given QoS class. The reason is that it's not possible to use IFH.vstax_avail = 1 when transmitting switched in a stacking configuration. QoS classification based on VLAN.PCP is enabled on the CPU port.

Validity: A F

```
Luton26: Y Y
Jaguar1: Y Y
Serval : Y Y
Jaguar2: Y Y
Serval2: Y Y
ServalT: Y Y
```

Definition at line 1649 of file vtss_packet_api.h.

7.104.2.7 tx_vstax_hdr

```
vtss_packet_tx_vstax_t vtss_packet_tx_info_t::tx_vstax_hdr
```

If a frame is going to be transmitted with a VStaX header, set this member to a value different from VTSS_PACKET_TX_VSTAX_NONE.

When transmitting with stack header, the application may choose to either provide a binary or a non-binary stack header to this function. The binary, selected with `tx_vstax_hdr == VTSS_PACKET_TX_VSTAX_BIN`, is useful if the application uses the same stack header in all frames it may send. Instead of encoding it every time, it may encode it once (with `vtss_packet_vstax_header2frame()`) and copy it to the bin member of `vstax` everytime it needs to send a frame with that stack header.

On the other hand, if the stack header changes from time to time, the application will probably wish to use VTSS_PACKET_TX_VSTAX_SYM, which causes the API to encode the stack header for it.

Validity: A F

```
Luton26: N N
Jaguar1: Y Y
Serval : N N
Jaguar2: Y Y
Serval2: N N
ServalT: N N
```

Definition at line 1679 of file vtss_packet_api.h.

7.104.2.8 bin

```
u8 vtss_packet_tx_info_t::bin[VTSS_VSTAX_HDR_SIZE]
```

This is the binary version of the VStaX header to insert when the frame gets transmitted on a stack port. It is only used if `tx_vstax_hdr` is `VTSS_PACKET_TX_VSTAX_BIN`. It may be composed with `vtss_packet_vstax_header2frame()`

Validity: A F

```
Luton26: N N
Jaguar1: Y Y
Serval : N N
Jaguar2: Y Y
Serval2: N N
ServalT: N N
```

Definition at line 1702 of file vtss_packet_api.h.

7.104.2.9 sym

```
vtss_vstax_tx_header_t vtss_packet_tx_info_t::sym
```

This is the symbolic version of the VStaX header to insert when the frame gets transmitted on a stack port. The binary version will be encoded [vtss_packet_tx_hdr_encode\(\)](#). It is only used if [tx_vstax_hdr](#) is [VTSS_PACKET_TX_VSTAX_SYM](#).

Validity: A F

```
Luton26: N N
Jaguar1: Y Y
Serval : N N
Jaguar2: Y Y
Serval2: N N
ServalT: N N
```

Definition at line 1721 of file vtss_packet_api.h.

7.104.2.10 vstax

```
union { ... } vtss_packet_tx_info_t::vstax
```

Contains the VStaX header in either binary or symbolic form. Contains the VStaX header in either binary or symbolic form.

7.104.2.11 ptp_action

```
vtss_packet_ptp_action_t vtss_packet_tx_info_t::ptp_action
```

The frame's Precision Time Protocol action. See [vtss_packet_ptp_action_t](#) for the enumeration. Ignored when [switch_frm](#) is TRUE.

When != [VTSS_PACKET_PTP_ACTION_NONE](#), the [ptp_timestamp](#) and [ptp_id](#) fields must be filled in.

Validity: A F

```
Luton26: Y Y - except for enumeration VTSS_PACKET_PTP_ACTION_ORIGIN_TIMESTAMP.
Jaguar1: N N
Serval : Y Y - except for enumeration VTSS_PACKET_PTP_ACTION_ONE_AND_TWO_STEP.
Jaguar2: Y Y - except for enumeration VTSS_PACKET_PTP_ACTION_ONE_AND_TWO_STEP.
Serval2: Y Y - except for enumeration VTSS_PACKET_PTP_ACTION_ONE_AND_TWO_STEP.
ServalT: Y Y - except for enumeration VTSS_PACKET_PTP_ACTION_ONE_AND_TWO_STEP.
```

Definition at line 1744 of file vtss_packet_api.h.

7.104.2.12 ptp_id

`u8 vtss_packet_tx_info_t::ptp_id`

The PTP identifier used for two-step PTP actions. The CPU can only use from IDs 0 through 3. Ignored when `switch_frm` is TRUE.

Used when `ptp_action` == VTSS_PACKET_PTP_ACTION_TWO_STEP.

Validity: A F

Luton26:	Y	Y
Jaguar1:	N	N
Serval :	Y	Y
Jaguar2:	Y	Y
Serval2:	Y	Y
ServalT:	Y	Y

Definition at line 1764 of file vtss_packet_api.h.

7.104.2.13 ptp_timestamp

`u32 vtss_packet_tx_info_t::ptp_timestamp`

Holds the PTP timestamp indicating when the injection started. The rewriter can then calculate a residence time based on this and the frame's transmission timestamp. Ignored when `switch_frm` is TRUE.

Used when `ptp_action` is != VTSS_PACKET_PTP_ACTION_NONE.

Validity: A F

Luton26:	Y	Y
Jaguar1:	N	N
Serval :	Y	Y
Jaguar2:	Y	Y
Serval2:	Y	Y
ServalT:	Y	Y

Definition at line 1785 of file vtss_packet_api.h.

7.104.2.14 latch_timestamp

`u32 vtss_packet_tx_info_t::latch_timestamp`

Latch timestamp into a switch core register when the frame is transmitted. This register can then be read-out by S/W to obtain the actual transmission time for that frame.

Encoding:

- 0: Don't latch timestamp.
- 1: Latch timestamp into register 0.
- 2: Latch timestamp into register 1.
- 3: Latch timestamp into register 2.

Validity: A F

Luton26: N N
Jaguar1: Y Y
Serval : N N
Jaguar2: N N
Serval2: N N
ServalT: N N

Definition at line 1810 of file vtss_packet_api.h.

7.104.2.15 oam_type

`vtss_packet_oam_type_t vtss_packet_tx_info_t::oam_type`

OAM type.

Only used if `ptp_action` is VTSS_PACKET_PTP_ACTION_NONE.

See `vtss_packet_oam_type_t` for a description. Ignored when `switch_frm` is TRUE.

Validity: A F

Luton26: N N
Jaguar1: N N
Serval : Y Y
Jaguar2: N N
Serval2: N N
ServalT: N N

Definition at line 1831 of file vtss_packet_api.h.

7.104.2.16 isdx

`vtss_isdx_t vtss_packet_tx_info_t::isdx`

Ingress Service Index.

If not VTSS_ISDX_NONE, it will be used in ES0 lookups instead of the frame's classified VID. See also [isdx_dont_use](#). Ignored when [switch_frm](#) is TRUE.

Validity: A F

Luton26: N N
Jaguar1: N N
Serval : Y Y
Jaguar2: N N
Serval2: N N
ServalT: N N

Definition at line 1852 of file `vtss_packet_api.h`.

7.104.2.17 isdx_dont_use

`BOOL vtss_packet_tx_info_t::isdx_dont_use`

When set to TRUE, [isdx](#) is not used for ES0 lookups, only for frame counting.

Ignored when [switch_frm](#) is TRUE or [isdx](#) is VTSS_ISDX_NONE.

Validity: A F

Luton26: N N
Jaguar1: N N
Jaguar2: N N
Serval : Y Y
Jaguar2: N N
Serval2: N N
ServalT: N N

Definition at line 1872 of file `vtss_packet_api.h`.

7.104.2.18 dp

`vtss_dp_level_t vtss_packet_tx_info_t::dp`

Drop Precedence.

The frame's drop precedence level after policing. Ignored when `switch_frm` is TRUE.

Validity: A F

Luton26: N N
Jaguar1: N N
Serval : Y Y
Jaguar2: N N
Serval2: N N
ServalT: N N

Definition at line 1891 of file vtss_packet_api.h.

7.104.2.19 masquerade_port

`vtss_port_no_t vtss_packet_tx_info_t::masquerade_port`

Masquerade port. When masquerading, the frame will be handled as if it was received by the ingress port specified in `masquerade_port`.

Its value will not be used unless `switch_frm` is TRUE. Also, when masquerading, the FDMA driver may or may not insert a VLAN tag into the frame prior to transmission. Please consult the FDMA driver API to locate the property that allows for insertion of a VLAN tag.

Set it to VTSS_PORT_NO_NONE to disable masquerading.

Validity: A F

Luton26: N N
Jaguar1: N N
Serval : Y Y
Jaguar2: N N
Serval2: N N
ServalT: N N

Definition at line 1916 of file vtss_packet_api.h.

7.104.2.20 pdu_offset

`u32 vtss_packet_tx_info_t::pdu_offset`

PDU offset in 8 bit word counts. Used in ptp-action's to indicate the start of the PTP PDU.

Validity: A F

```
Luton26: N N
Jaguar1: N N
Serval : N N
Jaguar2: Y Y
Serval2: Y Y
ServalT: Y Y
```

Definition at line 1933 of file `vtss_packet_api.h`.

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

- `vtss_api/include/vtss_packet_api.h`

7.105 vtss_phy_10g_apc_conf_t Struct Reference

10G Phy APC configuration

```
#include <vtss_phy_10g_api.h>
```

Data Fields

- `vtss_phy_10g_ib_apc_op_mode_t op_mode`
- `BOOL op_mode_flag`

7.105.1 Detailed Description

10G Phy APC configuration

Definition at line 241 of file `vtss_phy_10g_api.h`.

7.105.2 Field Documentation

7.105.2.1 op_mode

`vtss_phy_10g_ib_apc_op_mode_t vtss_phy_10g_apc_conf_t::op_mode`

APC operation

Definition at line 242 of file `vtss_phy_10g_api.h`.

7.105.2.2 op_mode_flag

`BOOL vtss_phy_10g_apc_conf_t::op_mode_flag`

APC operation flag,eg: TRUE= APC_RESET, FALSE = APC_RESET clear

Definition at line 243 of file `vtss_phy_10g_api.h`.

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

- `vtss_api/include/vtss_phy_10g_api.h`

7.106 vtss_phy_10g_apc_status_t Struct Reference

10G Phy APC status

```
#include <vtss_phy_10g_api.h>
```

Data Fields

- `BOOL reset`
- `BOOL freeze`

7.106.1 Detailed Description

10G Phy APC status

Definition at line 247 of file `vtss_phy_10g_api.h`.

7.106.2 Field Documentation

7.106.2.1 reset

`BOOL vtss_phy_10g_apc_status_t::reset`

APC reset status

Definition at line 248 of file `vtss_phy_10g_api.h`.

7.106.2.2 freeze

`BOOL vtss_phy_10g_apc_status_t::freeze`

APC freeze status

Definition at line 249 of file `vtss_phy_10g_api.h`.

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

- `vtss_api/include/vtss_phy_10g_api.h`

7.107 `vtss_phy_10g_auto_failover_conf_t` Struct Reference

10G PHY Automatic Failover configuration

```
#include <vtss_phy_10g_api.h>
```

Data Fields

- `vtss_port_no_t port_no`
- `vtss_phy_10g_auto_failover_event_t evnt`
- `u16 trig_ch_id`
- `BOOL is_host_side`
- `u16 channel_id`
- `vtss_gpio_10g_no_t v_gpio`
- `vtss_gpio_10g_no_t a_gpio`
- `BOOL enable`
- `vtss_phy_10g_auto_failover_filter_t filter`
- `u16 fltr_val`

7.107.1 Detailed Description

10G PHY Automatic Failover configuration

Definition at line 1807 of file `vtss_phy_10g_api.h`.

7.107.2 Field Documentation

7.107.2.1 port_no

`vtss_port_no_t vtss_phy_10g_auto_failover_conf_t::port_no`

port number

Definition at line 1808 of file vtss_phy_10g_api.h.

7.107.2.2 evnt

`vtss_phy_10g_auto_failover_event_t vtss_phy_10g_auto_failover_conf_t::evnt`

Auto failover event selection

Definition at line 1809 of file vtss_phy_10g_api.h.

7.107.2.3 trig_ch_id

`u16 vtss_phy_10g_auto_failover_conf_t::trig_ch_id`

Channel ID that triggers event,source

Definition at line 1810 of file vtss_phy_10g_api.h.

7.107.2.4 is_host_side

`BOOL vtss_phy_10g_auto_failover_conf_t::is_host_side`

Protection switch configuration is on line(RX) or host side(Rx)

Definition at line 1811 of file vtss_phy_10g_api.h.

7.107.2.5 channel_id

`u16 vtss_phy_10g_auto_failover_conf_t::channel_id`

channel to be switched,destination

Definition at line 1812 of file vtss_phy_10g_api.h.

7.107.2.6 v_gpio

`vtss_gpio_10g_no_t` `vtss_phy_10g_auto_failover_conf_t::v_gpio`

virtual GPIO pin to be triggering the event

Definition at line 1813 of file `vtss_phy_10g_api.h`.

7.107.2.7 a_gpio

`vtss_gpio_10g_no_t` `vtss_phy_10g_auto_failover_conf_t::a_gpio`

actual GPIO pin to be triggering the event

Definition at line 1814 of file `vtss_phy_10g_api.h`.

7.107.2.8 enable

`BOOL` `vtss_phy_10g_auto_failover_conf_t::enable`

Enable or disable auto failover

Definition at line 1815 of file `vtss_phy_10g_api.h`.

7.107.2.9 filter

`vtss_phy_10g_auto_failover_filter_t` `vtss_phy_10g_auto_failover_conf_t::filter`

Number of CSR clock cycles

Definition at line 1816 of file `vtss_phy_10g_api.h`.

7.107.2.10 fltr_val

`u16` `vtss_phy_10g_auto_failover_conf_t::fltr_val`

value of filter if chosen

Definition at line 1817 of file `vtss_phy_10g_api.h`.

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

- `vtss_api/include/vtss_phy_10g_api.h`

7.108 vtss_phy_10g_base_kr_autoneg_t Struct Reference

10G Phy Base KR Autoneg config

```
#include <vtss_phy_10g_api.h>
```

Data Fields

- `BOOL an_restart`
- `BOOL an_enable`
- `BOOL an_reset`

7.108.1 Detailed Description

10G Phy Base KR Autoneg config

Definition at line 1207 of file vtss_phy_10g_api.h.

7.108.2 Field Documentation

7.108.2.1 an_restart

```
BOOL vtss_phy_10g_base_kr_autoneg_t::an_restart
```

Autoneg restart (for debug)

Definition at line 1208 of file vtss_phy_10g_api.h.

7.108.2.2 an_enable

```
BOOL vtss_phy_10g_base_kr_autoneg_t::an_enable
```

Autoneg enable

Definition at line 1209 of file vtss_phy_10g_api.h.

7.108.2.3 an_reset

`BOOL vtss_phy_10g_base_kr_autoneg_t::an_reset`

Autoneg reset (for debug)

Definition at line 1210 of file `vtss_phy_10g_api.h`.

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

- `vtss_api/include/vtss_phy_10g_api.h`

7.109 `vtss_phy_10g_base_kr_conf_t` Struct Reference

10G Phy 10f_base_kr_conf config data according to 802.3-2008 clause 72.7 Figure 72-11

```
#include <vtss_phy_10g_api.h>
```

Data Fields

- `i32 cm1`
- `i32 c0`
- `i32 c1`
- `u32 ampl`
- `u32 slewrate`
- `BOOL en_ob`
- `BOOL ser_inv`

7.109.1 Detailed Description

10G Phy 10f_base_kr_conf config data according to 802.3-2008 clause 72.7 Figure 72-11

Definition at line 1195 of file `vtss_phy_10g_api.h`.

7.109.2 Field Documentation

7.109.2.1 cm1

`i32 vtss_phy_10g_base_kr_conf_t::cm1`

The minus 1 coefficient c(-1). Range: -32..31

Definition at line 1196 of file `vtss_phy_10g_api.h`.

7.109.2.2 c0

`i32 vtss_phy_10g_base_kr_conf_t::c0`

The 0 coefficient c(0). Range: -32..31

Definition at line 1197 of file vtss_phy_10g_api.h.

7.109.2.3 c1

`i32 vtss_phy_10g_base_kr_conf_t::c1`

The plus 1 coefficient c(1). Range: -32..31

Definition at line 1198 of file vtss_phy_10g_api.h.

7.109.2.4 ampl

`u32 vtss_phy_10g_base_kr_conf_t::ampl`

The Amplitude value in nVpp. Range: 300..1275

Definition at line 1199 of file vtss_phy_10g_api.h.

7.109.2.5 slewrate

`u32 vtss_phy_10g_base_kr_conf_t::slewrate`

Slew rate ctrl of OB

Definition at line 1200 of file vtss_phy_10g_api.h.

7.109.2.6 en_ob

`BOOL vtss_phy_10g_base_kr_conf_t::en_ob`

Enable output buffer and serializer

Definition at line 1201 of file vtss_phy_10g_api.h.

7.109.2.7 ser_inv

`BOOL vtss_phy_10g_base_kr_conf_t::ser_inv`

Invert input to serializer

Definition at line 1202 of file `vtss_phy_10g_api.h`.

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

- `vtss_api/include/vtss_phy_10g_api.h`

7.110 vtss_phy_10g_base_kr_ld_adv_abil_t Struct Reference

10G Phy Base Link Advertisement capability config

```
#include <vtss_phy_10g_api.h>
```

Data Fields

- `u8 adv_1g`
- `u8 adv_10g`
- `u8 fec_abil`
- `u8 fec_req`

7.110.1 Detailed Description

10G Phy Base Link Advertisement capability config

Definition at line 1223 of file `vtss_phy_10g_api.h`.

7.110.2 Field Documentation

7.110.2.1 adv_1g

`u8 vtss_phy_10g_base_kr_ld_adv_abil_t::adv_1g`

Advertise 1G ,not supported

Definition at line 1224 of file `vtss_phy_10g_api.h`.

7.110.2.2 adv_10g

`u8 vtss_phy_10g_base_kr_ld_adv_abil_t::adv_10g`

Advertise 10G,by default choosen by API

Definition at line 1225 of file vtss_phy_10g_api.h.

7.110.2.3 fec_abil

`u8 vtss_phy_10g_base_kr_ld_adv_abil_t::fec_abil`

Set FEC ability,by default choosen by API

Definition at line 1226 of file vtss_phy_10g_api.h.

7.110.2.4 fec_req

`u8 vtss_phy_10g_base_kr_ld_adv_abil_t::fec_req`

Set FEC request ,the only configurable option

Definition at line 1227 of file vtss_phy_10g_api.h.

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

- vtss_api/include/vtss_phy_10g_api.h

7.111 vtss_phy_10g_base_kr_status_t Struct Reference

10G Phy Base KR Training & Autoneg status

```
#include <vtss_phy_10g_api.h>
```

Data Fields

- [vtss_phy_10g_kr_status_aneg_t aneg](#)
- [vtss_phy_10g_kr_status_train_t train](#)
- [vtss_phy_10g_kr_status_fec_t fec](#)

7.111.1 Detailed Description

10G Phy Base KR Training & Autoneg status

Definition at line 1268 of file vtss_phy_10g_api.h.

7.111.2 Field Documentation

7.111.2.1 aneg

`vtss_phy_10g_kr_status_aneg_t vtss_phy_10g_base_kr_status_t::aneg`

Aneg structure

Definition at line 1269 of file `vtss_phy_10g_api.h`.

7.111.2.2 train

`vtss_phy_10g_kr_status_train_t vtss_phy_10g_base_kr_status_t::train`

Training structure

Definition at line 1270 of file `vtss_phy_10g_api.h`.

7.111.2.3 fec

`vtss_phy_10g_kr_status_fec_t vtss_phy_10g_base_kr_status_t::fec`

FEC structure

Definition at line 1271 of file `vtss_phy_10g_api.h`.

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

- `vtss_api/include/vtss_phy_10g_api.h`

7.112 `vtss_phy_10g_base_kr_train_aneg_t` Struct Reference

10G Phy Base KR Training & Autoneg config

```
#include <vtss_phy_10g_api.h>
```

Data Fields

- `vtss_phy_10g_base_kr_training_t training`
- `vtss_phy_10g_base_kr_autoneg_t autoneg`
- `vtss_phy_10g_base_kr_id_adv_abil_t id_abil`
- `BOOL host_kr`
- `BOOL line_kr`

7.112.1 Detailed Description

10G Phy Base KR Training & Autoneg config

Definition at line 1231 of file vtss_phy_10g_api.h.

7.112.2 Field Documentation

7.112.2.1 training

```
vtss_phy_10g_base_kr_training_t vtss_phy_10g_base_kr_train_aneg_t::training
```

KR Training params

Definition at line 1232 of file vtss_phy_10g_api.h.

7.112.2.2 autoneg

```
vtss_phy_10g_base_kr_autoneg_t vtss_phy_10g_base_kr_train_aneg_t::autoneg
```

KR Autoneg params

Definition at line 1233 of file vtss_phy_10g_api.h.

7.112.2.3 ld_abil

```
vtss_phy_10g_base_kr_ld_adv_abil_t vtss_phy_10g_base_kr_train_aneg_t::ld_abil
```

KR LD ADV Ability params

Definition at line 1234 of file vtss_phy_10g_api.h.

7.112.2.4 host_kr

```
BOOL vtss_phy_10g_base_kr_train_aneg_t::host_kr
```

Host side KR operation

Definition at line 1235 of file vtss_phy_10g_api.h.

7.112.2.5 line_kr

`BOOL vtss_phy_10g_base_kr_train_aneg_t::line_kr`

Line side KR operation

Definition at line 1236 of file `vtss_phy_10g_api.h`.

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

- `vtss_api/include/vtss_phy_10g_api.h`

7.113 `vtss_phy_10g_base_kr_training_t` Struct Reference

10G Phy Base KR Training config

```
#include <vtss_phy_10g_api.h>
```

Data Fields

- `BOOL enable`
- `u8 trmthd_cp`
- `u8 trmthd_c0`
- `u8 trmthd_cm`
- `BOOL id_pre_init`

7.113.1 Detailed Description

10G Phy Base KR Training config

Definition at line 1214 of file `vtss_phy_10g_api.h`.

7.113.2 Field Documentation

7.113.2.1 enable

`BOOL vtss_phy_10g_base_kr_training_t::enable`

Enable KR training

Definition at line 1215 of file `vtss_phy_10g_api.h`.

7.113.2.2 trmthd_cp

`u8 vtss_phy_10g_base_kr_training_t::trmthd_cp`

Training method c(+1), 0-BER is supported

Definition at line 1216 of file `vtss_phy_10g_api.h`.

7.113.2.3 trmthd_c0

`u8 vtss_phy_10g_base_kr_training_t::trmthd_c0`

Training method c(0) , 0-BER,1-GAIN are supported

Definition at line 1217 of file `vtss_phy_10g_api.h`.

7.113.2.4 trmthd_cm

`u8 vtss_phy_10g_base_kr_training_t::trmthd_cm`

Training method c(-1), 0-BER is supported

Definition at line 1218 of file `vtss_phy_10g_api.h`.

7.113.2.5 ld_pre_init

`BOOL vtss_phy_10g_base_kr_training_t::ld_pre_init`

Set local taps starting point 0-initialize,1-preset

Definition at line 1219 of file `vtss_phy_10g_api.h`.

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

- `vtss_api/include/vtss_phy_10g_api.h`

7.114 **vtss_phy_10g_ckout_conf_t** Struct Reference

10G Phy CKOUT config data

```
#include <vtss_phy_10g_api.h>
```

Data Fields

- `vtss_ckout_data_sel_t mode`
- `vtss_phy_10g_squelch_src_t src`
- `vtss_phy_10g_ckout_freq_t freq`
- `BOOL squelch_inv`
- `BOOL enable`
- `ckout_sel_t ckout_sel`

7.114.1 Detailed Description

10G Phy CKOUT config data

Malibu Only

Definition at line 962 of file `vtss_phy_10g_api.h`.

7.114.2 Field Documentation

7.114.2.1 mode

`vtss_ckout_data_sel_t vtss_phy_10g_ckout_conf_t::mode`

CKOUT output clock mode

Definition at line 963 of file `vtss_phy_10g_api.h`.

7.114.2.2 src

`vtss_phy_10g_squelch_src_t vtss_phy_10g_ckout_conf_t::src`

CKOUT squelch source

Definition at line 964 of file `vtss_phy_10g_api.h`.

7.114.2.3 freq

`vtss_phy_10g_ckout_freq_t vtss_phy_10g_ckout_conf_t::freq`

CKOUT clock frequency

Definition at line 965 of file `vtss_phy_10g_api.h`.

7.114.2.4 squelch_inv

`BOOL vtss_phy_10g_ckout_conf_t::squelch_inv`

'0'- Use squelch source src as is, '1'-Invert

Definition at line 966 of file vtss_phy_10g_api.h.

7.114.2.5 enable

`BOOL vtss_phy_10g_ckout_conf_t::enable`

'1'- Enable CKOUT, '0'-Disable

Definition at line 967 of file vtss_phy_10g_api.h.

7.114.2.6 ckout_sel

`ckout_sel_t vtss_phy_10g_ckout_conf_t::ckout_sel`

CKOUT sel eg-'0' for CKOUT0, '1' for CKOUT1

Definition at line 968 of file vtss_phy_10g_api.h.

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

- vtss_api/include/[vtss_phy_10g_api.h](#)

7.115 vtss_phy_10g_clause_37_adv_t Struct Reference

Advertisement control data for Clause 37 aneg.

```
#include <vtss_phy_10g_api.h>
```

Data Fields

- `BOOL fdx`
- `BOOL hdx`
- `BOOL symmetric_pause`
- `BOOL asymmetric_pause`
- `vtss_phy_10g_clause_37_remote_fault_t remote_fault`
- `BOOL acknowledge`
- `BOOL next_page`

7.115.1 Detailed Description

Advertisement control data for Clause 37 aneg.

Definition at line 1507 of file vtss_phy_10g_api.h.

7.115.2 Field Documentation

7.115.2.1 fdx

`BOOL vtss_phy_10g_clause_37_adv_t::fdx`

(FD)

Definition at line 1509 of file vtss_phy_10g_api.h.

7.115.2.2 hdx

`BOOL vtss_phy_10g_clause_37_adv_t::hdx`

(HD) ,Not supported

Definition at line 1510 of file vtss_phy_10g_api.h.

7.115.2.3 symmetric_pause

`BOOL vtss_phy_10g_clause_37_adv_t::symmetric_pause`

(PS1)

Definition at line 1511 of file vtss_phy_10g_api.h.

7.115.2.4 asymmetric_pause

`BOOL vtss_phy_10g_clause_37_adv_t::asymmetric_pause`

(PS2)

Definition at line 1512 of file vtss_phy_10g_api.h.

7.115.2.5 remote_fault

`vtss_phy_10g_clause_37_remote_fault_t vtss_phy_10g_clause_37_adv_t::remote_fault`

(RF1) + (RF2) , would be generated according to condition

Definition at line 1513 of file vtss_phy_10g_api.h.

7.115.2.6 acknowledge

`BOOL vtss_phy_10g_clause_37_adv_t::acknowledge`

(Ack) , would be generated according to condition

Definition at line 1514 of file vtss_phy_10g_api.h.

7.115.2.7 next_page

`BOOL vtss_phy_10g_clause_37_adv_t::next_page`

(NP) ,Not supported

Definition at line 1515 of file vtss_phy_10g_api.h.

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

- `vtss_api/include/vtss_phy_10g_api.h`

7.116 vtss_phy_10g_clause_37_cmn_status_t Struct Reference

Clause 37 Auto-negotiation status for line and host.

```
#include <vtss_phy_10g_api.h>
```

Data Fields

- `vtss_phy_10g_clause_37_status_t line`
- `vtss_phy_10g_clause_37_status_t host`

7.116.1 Detailed Description

Clause 37 Auto-negotiation status for line and host.

Definition at line 1529 of file vtss_phy_10g_api.h.

7.116.2 Field Documentation

7.116.2.1 line

`vtss_phy_10g_clause_37_status_t vtss_phy_10g_clause_37_cmn_status_t::line`

Line clause 37 status

Definition at line 1531 of file `vtss_phy_10g_api.h`.

7.116.2.2 host

`vtss_phy_10g_clause_37_status_t vtss_phy_10g_clause_37_cmn_status_t::host`

Host clause 37 status

Definition at line 1532 of file `vtss_phy_10g_api.h`.

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

- `vtss_api/include/vtss_phy_10g_api.h`

7.117 `vtss_phy_10g_clause_37_control_t` Struct Reference

Clause 37 control struct.

```
#include <vtss_phy_10g_api.h>
```

Data Fields

- `BOOL enable`
- `vtss_phy_10g_clause_37_adv_t advertisement`
- `BOOL enable_pass_thru`
- `BOOL line`
- `BOOL host`
- `BOOL l_h`

7.117.1 Detailed Description

Clause 37 control struct.

Definition at line 1537 of file `vtss_phy_10g_api.h`.

7.117.2 Field Documentation

7.117.2.1 enable

`BOOL vtss_phy_10g_clause_37_control_t::enable`

Enable of Autoneg

Definition at line 1539 of file vtss_phy_10g_api.h.

7.117.2.2 advertisement

`vtss_phy_10g_clause_37_adv_t vtss_phy_10g_clause_37_control_t::advertisement`

Clause 37 Advertisement data

Definition at line 1540 of file vtss_phy_10g_api.h.

7.117.2.3 enable_pass_thru

`BOOL vtss_phy_10g_clause_37_control_t::enable_pass_thru`

Enables pass through mode in VENICE/MALIBU

Definition at line 1541 of file vtss_phy_10g_api.h.

7.117.2.4 line

`BOOL vtss_phy_10g_clause_37_control_t::line`

Line:TRUE for line side

Definition at line 1542 of file vtss_phy_10g_api.h.

7.117.2.5 host

`BOOL vtss_phy_10g_clause_37_control_t::host`

Host:True for host side

Definition at line 1543 of file vtss_phy_10g_api.h.

7.117.2.6 l_h

`BOOL vtss_phy_10g_clause_37_control_t::l_h`

Both Host and line side

Definition at line 1544 of file vtss_phy_10g_api.h.

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

- `vtss_api/include/vtss_phy_10g_api.h`

7.118 vtss_phy_10g_clause_37_status_t Struct Reference

Clause 37 Auto-negotiation status.

```
#include <vtss_phy_10g_api.h>
```

Data Fields

- `BOOL link`
- `struct {
 BOOL complete
 vtss_phy_10g_clause_37_adv_t partner_advertisement
} autoneg`

7.118.1 Detailed Description

Clause 37 Auto-negotiation status.

Definition at line 1519 of file vtss_phy_10g_api.h.

7.118.2 Field Documentation

7.118.2.1 link

`BOOL vtss_phy_10g_clause_37_status_t::link`

FALSE if link has been down since last status read

Definition at line 1521 of file vtss_phy_10g_api.h.

7.118.2.2 complete

`BOOL vtss_phy_10g_clause_37_status_t::complete`

Aneg completion status

Definition at line 1523 of file vtss_phy_10g_api.h.

7.118.2.3 partner_advertisement

`vtss_phy_10g_clause_37_adv_t vtss_phy_10g_clause_37_status_t::partner_advertisement`

Clause 37 Advertisement control data

Definition at line 1524 of file vtss_phy_10g_api.h.

7.118.2.4 autoneg

`struct { ... } vtss_phy_10g_clause_37_status_t::autoneg`

Autoneg status

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

- `vtss_api/include/vtss_phy_10g_api.h`

7.119 vtss_phy_10g_clk_src_t Struct Reference

10G Phy CLOCK Source Selection

```
#include <vtss_phy_10g_api.h>
```

Data Fields

- `BOOL is_high_amp`

7.119.1 Detailed Description

10G Phy CLOCK Source Selection

Definition at line 192 of file vtss_phy_10g_api.h.

7.119.2 Field Documentation

7.119.2.1 is_high_amp

`BOOL vtss_phy_10g_clk_src_t::is_high_amp`

Amplitude selection HIGH or LOW

Definition at line 193 of file `vtss_phy_10g_api.h`.

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

- `vtss_api/include/vtss_phy_10g_api.h`

7.120 vtss_phy_10g_cnt_t Struct Reference

10G Phy Sublayer counters

```
#include <vtss_phy_10g_api.h>
```

Data Fields

- `vtss_phy_pcs_cnt_t pcs`

7.120.1 Detailed Description

10G Phy Sublayer counters

Definition at line 1676 of file `vtss_phy_10g_api.h`.

7.120.2 Field Documentation

7.120.2.1 pcs

`vtss_phy_pcs_cnt_t vtss_phy_10g_cnt_t::pcs`

PCS counters

Definition at line 1679 of file `vtss_phy_10g_api.h`.

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

- `vtss_api/include/vtss_phy_10g_api.h`

7.121 vtss_phy_10g_fifo_sync_t Struct Reference

10G OOS workaround options

```
#include <vtss_phy_ts_api.h>
```

Data Fields

- `BOOL bypass_in_api`
- `BOOL skip_rev_check`
- `vtss_debug_printf_t pr`

7.121.1 Detailed Description

10G OOS workaround options

Definition at line 2090 of file vtss_phy_ts_api.h.

7.121.2 Field Documentation

7.121.2.1 bypass_in_api

```
BOOL vtss_phy_10g_fifo_sync_t::bypass_in_api
```

clear bypass in API

Definition at line 2091 of file vtss_phy_ts_api.h.

7.121.2.2 skip_rev_check

```
BOOL vtss_phy_10g_fifo_sync_t::skip_rev_check
```

To force execution irrespective of revision

Definition at line 2092 of file vtss_phy_ts_api.h.

7.121.2.3 pr

```
vtss_debug_printf_t vtss_phy_10g_fifo_sync_t::pr
```

Pass print function to get the algorithm execution logs

Definition at line 2093 of file vtss_phy_ts_api.h.

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

- vtss_api/include/[vtss_phy_ts_api.h](#)

7.122 vtss_phy_10g_fw_status_t Struct Reference

Firmware status.

```
#include <vtss_phy_10g_api.h>
```

Data Fields

- [u16 edc_fw_rev](#)
- [BOOL edc_fw_chksum](#)
- [BOOL icpu_activity](#)
- [BOOL edc_fw_api_load](#)

7.122.1 Detailed Description

Firmware status.

Definition at line 2741 of file vtss_phy_10g_api.h.

7.122.2 Field Documentation

7.122.2.1 edc_fw_rev

```
u16 vtss_phy_10g_fw_status_t::edc_fw_rev
```

FW revision

Definition at line 2742 of file vtss_phy_10g_api.h.

7.122.2.2 edc_fw_chksum

`BOOL vtss_phy_10g_fw_status_t::edc_fw_chksum`

FW checksum. Fail=0, Pass=1

Definition at line 2743 of file vtss_phy_10g_api.h.

7.122.2.3 icpu_activity

`BOOL vtss_phy_10g_fw_status_t::icpu_activity`

iCPU activity. Not Running=0, Running=1

Definition at line 2744 of file vtss_phy_10g_api.h.

7.122.2.4 edc_fw_api_load

`BOOL vtss_phy_10g_fw_status_t::edc_fw_api_load`

EDC FW is loaded through API No=0, Yes=1

Definition at line 2745 of file vtss_phy_10g_api.h.

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

- `vtss_api/include/vtss_phy_10g_api.h`

7.123 vtss_phy_10g_host_clk_conf_t Struct Reference

10G Phy Host clock config data

```
#include <vtss_phy_10g_api.h>
```

Data Fields

- `vtss_phy_10g_clk_sel_t mode`
- `vtss_phy_10g_recvrd_clk_sel_t recvrd_clk_sel`
- `u8 clk_sel_no`

7.123.1 Detailed Description

10G Phy Host clock config data

Malibu Only

Definition at line 1052 of file vtss_phy_10g_api.h.

7.123.2 Field Documentation

7.123.2.1 mode

`vtss_phy_10g_clk_sel_t` `vtss_phy_10g_host_clk_conf_t::mode`

Host side output clock mode

Definition at line 1053 of file vtss_phy_10g_api.h.

7.123.2.2 recvrd_clk_sel

`vtss_phy_10g_recvrd_clk_sel_t` `vtss_phy_10g_host_clk_conf_t::recvrd_clk_sel`

Recovered clock selection

Definition at line 1054 of file vtss_phy_10g_api.h.

7.123.2.3 clk_sel_no

`u8` `vtss_phy_10g_host_clk_conf_t::clk_sel_no`

Host clock select No(0-3)

Definition at line 1055 of file vtss_phy_10g_api.h.

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

- `vtss_api/include/vtss_phy_10g_api.h`

7.124 vtss_phy_10g_ib_conf_t Struct Reference

10G Phy IB configuration

```
#include <vtss_phy_10g_api.h>
```

Data Fields

- `ib_par_cfg offs`
- `ib_par_cfg gain`
- `ib_par_cfg gainadj`
- `ib_par_cfg l`
- `ib_par_cfg c`
- `ib_par_cfg agc`
- `ib_par_cfg dfe1`
- `ib_par_cfg dfe2`
- `ib_par_cfg dfe3`
- `ib_par_cfg dfe4`
- `u8 ld`
- `u8 prbs`
- `BOOL prbs_inv`
- `u32 apc_bit_mask`
- `u32 freeze_bit_mask`
- `u32 config_bit_mask`
- `BOOL is_host`

7.124.1 Detailed Description

10G Phy IB configuration

Definition at line 213 of file vtss_phy_10g_api.h.

7.124.2 Field Documentation

7.124.2.1 offs

`ib_par_cfg vtss_phy_10g_ib_conf_t::offs`

Equalizer offset value

Definition at line 214 of file vtss_phy_10g_api.h.

7.124.2.2 gain

`ib_par_cfg vtss_phy_10g_ib_conf_t::gain`

Equalizer gain value

Definition at line 215 of file vtss_phy_10g_api.h.

7.124.2.3 gainadj

`ib_par_cfg vtss_phy_10g_ib_conf_t::gainadj`

IB gain adjustment

Definition at line 216 of file vtss_phy_10g_api.h.

7.124.2.4 l

`ib_par_cfg vtss_phy_10g_ib_conf_t::l`

Equalizer L value

Definition at line 217 of file vtss_phy_10g_api.h.

7.124.2.5 c

`ib_par_cfg vtss_phy_10g_ib_conf_t::c`

Equalizer C value

Definition at line 218 of file vtss_phy_10g_api.h.

7.124.2.6 agc

`ib_par_cfg vtss_phy_10g_ib_conf_t::agc`

AGC value

Definition at line 219 of file vtss_phy_10g_api.h.

7.124.2.7 dfe1

`ib_par_cfg vtss_phy_10g_ib_conf_t::dfe1`

DFE1 active value

Definition at line 220 of file vtss_phy_10g_api.h.

7.124.2.8 dfe2

`ib_par_cfg` vtss_phy_10g_ib_conf_t::dfe2

DFE2 active value

Definition at line 221 of file vtss_phy_10g_api.h.

7.124.2.9 dfe3

`ib_par_cfg` vtss_phy_10g_ib_conf_t::dfe3

DFE3 active value

Definition at line 222 of file vtss_phy_10g_api.h.

7.124.2.10 dfe4

`ib_par_cfg` vtss_phy_10g_ib_conf_t::dfe4

DFE4 active value

Definition at line 223 of file vtss_phy_10g_api.h.

7.124.2.11 ld

`u8` vtss_phy_10g_ib_conf_t::ld

level detect

Definition at line 224 of file vtss_phy_10g_api.h.

7.124.2.12 prbs

`u8` vtss_phy_10g_ib_conf_t::prbs

PRBS RX pattern selected

Definition at line 225 of file vtss_phy_10g_api.h.

7.124.2.13 prbs_inv

`BOOL vtss_phy_10g_ib_conf_t::prbs_inv`

PRBS inversions selected

Definition at line 226 of file `vtss_phy_10g_api.h`.

7.124.2.14 apc_bit_mask

`u32 vtss_phy_10g_ib_conf_t::apc_bit_mask`

Bit mask that has the information of the all the parameters whether they are being controlled by APC

Definition at line 227 of file `vtss_phy_10g_api.h`.

7.124.2.15 freeze_bit_mask

`u32 vtss_phy_10g_ib_conf_t::freeze_bit_mask`

Bit mask that has the information of the all parameters that are frozen to the value

Definition at line 228 of file `vtss_phy_10g_api.h`.

7.124.2.16 config_bit_mask

`u32 vtss_phy_10g_ib_conf_t::config_bit_mask`

Bit mask that has the information of the all parameters that are to be configured

Definition at line 229 of file `vtss_phy_10g_api.h`.

7.124.2.17 is_host

`BOOL vtss_phy_10g_ib_conf_t::is_host`

Configuration is on Host or line

Definition at line 230 of file `vtss_phy_10g_api.h`.

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

- `vtss_api/include/vtss_phy_10g_api.h`

7.125 vtss_phy_10g_ib_status_t Struct Reference

10G Phy IB configuration

```
#include <vtss_phy_10g_api.h>
```

Data Fields

- [vtss_phy_10g_ib_conf_t ib_conf](#)
- [BOOL sig_det](#)
- [u16 bit_errors](#)

7.125.1 Detailed Description

10G Phy IB configuration

Definition at line 234 of file vtss_phy_10g_api.h.

7.125.2 Field Documentation

7.125.2.1 ib_conf

```
vtss\_phy\_10g\_ib\_conf\_t vtss_phy_10g_ib_status_t::ib_conf
```

Current status of IB configuraion

Definition at line 235 of file vtss_phy_10g_api.h.

7.125.2.2 sig_det

```
BOOL vtss_phy_10g_ib_status_t::sig_det
```

Signal detect

Definition at line 236 of file vtss_phy_10g_api.h.

7.125.2.3 bit_errors

```
u16 vtss_phy_10g_ib_status_t::bit_errors
```

Bit errors if PRBS is enabled

Definition at line 237 of file vtss_phy_10g_api.h.

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

- vtss_api/include/[vtss_phy_10g_api.h](#)

7.126 vtss_phy_10g_ib_storage_t Struct Reference

VSCOPE fast scan storage.

```
#include <vtss_phy_10g_api.h>
```

Data Fields

- `BOOL ib_storage_bool [BOOLEAN_STORAGE_COUNT]`
- `u32 ib_storage [UNSIGNED_STORAGE_COUNT]`

7.126.1 Detailed Description

VSCOPE fast scan storage.

Definition at line 1898 of file vtss_phy_10g_api.h.

7.126.2 Field Documentation

7.126.2.1 ib_storage_bool

```
BOOL vtss_phy_10g_ib_storage_t::ib_storage_bool [BOOLEAN_STORAGE_COUNT]
```

boolean values to be stored in vtss_state during vscope fast scan configuration

Definition at line 1899 of file vtss_phy_10g_api.h.

7.126.2.2 ib_storage

```
u32 vtss_phy_10g_ib_storage_t::ib_storage[UNSIGNED_STORAGE_COUNT]
```

u8 values to be stored in vtss_state during vscope fast scan configuration

Definition at line 1900 of file vtss_phy_10g_api.h.

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

- vtss_api/include/[vtss_phy_10g_api.h](#)

7.127 vtss_phy_10g_id_t Struct Reference

10G Phy part number and revision

```
#include <vtss_phy_10g_api.h>
```

Data Fields

- [u16 part_number](#)
- [u16 revision](#)
- [u16 channel_id](#)
- [vtss_phy_10g_family_t family](#)
- [vtss_phy_10g_type_t type](#)
- [vtss_port_no_t phy_api_base_no](#)
- [u16 device_feature_status](#)

7.127.1 Detailed Description

10G Phy part number and revision

Definition at line 2269 of file vtss_phy_10g_api.h.

7.127.2 Field Documentation

7.127.2.1 part_number

```
u16 vtss_phy_10g_id_t::part_number
```

Part number (Hex)

Definition at line 2271 of file vtss_phy_10g_api.h.

7.127.2.2 revision

`u16 vtss_phy_10g_id_t::revision`

Chip revision

Definition at line 2272 of file vtss_phy_10g_api.h.

7.127.2.3 channel_id

`u16 vtss_phy_10g_id_t::channel_id`

Channel id

Definition at line 2273 of file vtss_phy_10g_api.h.

7.127.2.4 family

`vtss_phy_10g_family_t vtss_phy_10g_id_t::family`

Phy Family

Definition at line 2274 of file vtss_phy_10g_api.h.

7.127.2.5 type

`vtss_phy_10g_type_t vtss_phy_10g_id_t::type`

Phy id (Decimal)

Definition at line 2275 of file vtss_phy_10g_api.h.

7.127.2.6 phy_api_base_no

`vtss_port_no_t vtss_phy_10g_id_t::phy_api_base_no`

First API no within this phy (in case of multiple channels)

Definition at line 2276 of file vtss_phy_10g_api.h.

7.127.2.7 device_feature_status

`u16 vtss_phy_10g_id_t::device_feature_status`

Device features depending on EFUSE

Definition at line 2277 of file vtss_phy_10g_api.h.

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

- `vtss_api/include/vtss_phy_10g_api.h`

7.128 vtss_phy_10g_init_parm_t Struct Reference

10G Phy Initialization configuration

```
#include <vtss_phy_10g_api.h>
```

Data Fields

- `vtss_channel_t channel_conf`

7.128.1 Detailed Description

10G Phy Initialization configuration

Definition at line 432 of file vtss_phy_10g_api.h.

7.128.2 Field Documentation

7.128.2.1 channel_conf

`vtss_channel_t vtss_phy_10g_init_parm_t::channel_conf`

Channel configuration selection,manual or auto

Definition at line 433 of file vtss_phy_10g_api.h.

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

- `vtss_api/include/vtss_phy_10g_api.h`

7.129 vtss_phy_10g_jitter_conf_t Struct Reference

10G Phy Optimisation of jitter performance

```
#include <vtss_phy_10g_api.h>
```

Data Fields

- `BOOL incr_levn`
- `u8 levn`
- `u8 vtail`

7.129.1 Detailed Description

10G Phy Optimisation of jitter performance

Definition at line 302 of file vtss_phy_10g_api.h.

7.129.2 Field Documentation

7.129.2.1 incr_levn

```
BOOL vtss_phy_10g_jitter_conf_t::incr_levn
```

Increase LevN

Definition at line 303 of file vtss_phy_10g_api.h.

7.129.2.2 levn

```
u8 vtss_phy_10g_jitter_conf_t::levn
```

Selects levn value depending on incr_levn value
incr_levn=1: levn: it is from 31: ~300mVpp to 0: ~1075mVpp.
incr_levn=0: levn: it is from 31: ~500mVpp to 0: ~1275mVpp. Maximum achievable amplitude depends on supply Voltage
Recommended settings for SR at 10.3125Gbps For Insertion loss < 4db Levn: 7 Incr_levn: 1 (Also the API Default)
Recommended settings for DAC Irrespective of Insertion loss Levn: 7 Incr_levn: 1 (Also the API Default)

Definition at line 304 of file vtss_phy_10g_api.h.

7.129.2.3 vtail

```
u8 vtss_phy_10g_jitter_conf_t::vtail
```

Vtail configuration 0: reserved, 1: 75mV, 2:100mV. Recommended settings(default in API) SR mode(@10.3125← Gbps), insertion loss < 4dB, value for vtail: 2 DAC mode, insertion loss independent,value for vtail: 2

Definition at line 314 of file vtss_phy_10g_api.h.

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

- vtss_api/include/[vtss_phy_10g_api.h](#)

7.130 vtss_phy_10g_kr_status_aneg_t Struct Reference

10G Phy Base KR Autoneg status

```
#include <vtss_phy_10g_api.h>
```

Data Fields

- [BOOL complete](#)
- [BOOL active](#)
- [BOOL request_10g](#)
- [BOOL request_1g](#)
- [BOOL request_fec_change](#)
- [BOOL fec_enable](#)
- [u32 sm](#)
- [BOOL lp_aneg_able](#)
- [BOOL block_lock](#)

7.130.1 Detailed Description

10G Phy Base KR Autoneg status

Definition at line 1240 of file vtss_phy_10g_api.h.

7.130.2 Field Documentation

7.130.2.1 complete

```
BOOL vtss_phy_10g_kr_status_aneg_t::complete
```

Aneg completed successfully

Definition at line 1241 of file vtss_phy_10g_api.h.

7.130.2.2 active

```
BOOL vtss_phy_10g_kr_status_aneg_t::active
```

Aneg is running between LD and LP

Definition at line 1242 of file vtss_phy_10g_api.h.

7.130.2.3 request_10g

```
BOOL vtss_phy_10g_kr_status_aneg_t::request_10g
```

LP's 10G rate negotiated

Definition at line 1243 of file vtss_phy_10g_api.h.

7.130.2.4 request_1g

```
BOOL vtss_phy_10g_kr_status_aneg_t::request_1g
```

LP's 1G rate negotiated

Definition at line 1244 of file vtss_phy_10g_api.h.

7.130.2.5 request_fec_change

```
BOOL vtss_phy_10g_kr_status_aneg_t::request_fec_change
```

LP's FEC request

Definition at line 1245 of file vtss_phy_10g_api.h.

7.130.2.6 fec_enable

```
BOOL vtss_phy_10g_kr_status_aneg_t::fec_enable
```

LP's FEC ability

Definition at line 1246 of file vtss_phy_10g_api.h.

7.130.2.7 sm

`u32 vtss_phy_10g_kr_status_aneg_t::sm`

Aneg state machine(debug)

Definition at line 1247 of file vtss_phy_10g_api.h.

7.130.2.8 lp_aneg_able

`BOOL vtss_phy_10g_kr_status_aneg_t::lp_aneg_able`

Link partner(LP) aneg ability

Definition at line 1248 of file vtss_phy_10g_api.h.

7.130.2.9 block_lock

`BOOL vtss_phy_10g_kr_status_aneg_t::block_lock`

PCS block lock

Definition at line 1249 of file vtss_phy_10g_api.h.

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

- `vtss_api/include/vtss_phy_10g_api.h`

7.131 vtss_phy_10g_kr_status_fec_t Struct Reference

10G Phy Base KR FEC status

```
#include <vtss_phy_10g_api.h>
```

Data Fields

- `BOOL enable`
- `u32 corrected_block_cnt`
- `u32 uncorrected_block_cnt`

7.131.1 Detailed Description

10G Phy Base KR FEC status

Definition at line 1261 of file vtss_phy_10g_api.h.

7.131.2 Field Documentation

7.131.2.1 enable

`BOOL vtss_phy_10g_kr_status_fec_t::enable`

FEC Enabled

Definition at line 1262 of file `vtss_phy_10g_api.h`.

7.131.2.2 corrected_block_cnt

`u32 vtss_phy_10g_kr_status_fec_t::corrected_block_cnt`

Corrected block count

Definition at line 1263 of file `vtss_phy_10g_api.h`.

7.131.2.3 uncorrected_block_cnt

`u32 vtss_phy_10g_kr_status_fec_t::uncorrected_block_cnt`

Un-corrected block count

Definition at line 1264 of file `vtss_phy_10g_api.h`.

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

- `vtss_api/include/vtss_phy_10g_api.h`

7.132 vtss_phy_10g_kr_status_train_t Struct Reference

10G Phy Base KR Training status

```
#include <vtss_phy_10g_api.h>
```

Data Fields

- `BOOL complete`
- `u8 cm_ob_tap_result`
- `u8 cp_ob_tap_result`
- `u8 c0_ob_tap_result`

7.132.1 Detailed Description

10G Phy Base KR Training status

Definition at line 1253 of file vtss_phy_10g_api.h.

7.132.2 Field Documentation

7.132.2.1 complete

```
BOOL vtss_phy_10g_kr_status_train_t::complete
```

Training completed successfully

Definition at line 1254 of file vtss_phy_10g_api.h.

7.132.2.2 cm_ob_tap_result

```
u8 vtss_phy_10g_kr_status_train_t::cm_ob_tap_result
```

The minus 1 coefficient c(-1). Range: -32..31

Definition at line 1255 of file vtss_phy_10g_api.h.

7.132.2.3 cp_ob_tap_result

```
u8 vtss_phy_10g_kr_status_train_t::cp_ob_tap_result
```

The 0 coefficient c(0). Range: -32..31

Definition at line 1256 of file vtss_phy_10g_api.h.

7.132.2.4 c0_ob_tap_result

```
u8 vtss_phy_10g_kr_status_train_t::c0_ob_tap_result
```

The plus 1 coefficient c(1). Range: -32..31

Definition at line 1257 of file vtss_phy_10g_api.h.

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

- vtss_api/include/[vtss_phy_10g_api.h](#)

7.133 vtss_phy_10g_lane_sync_conf_t Struct Reference

10G Phy Lane SYNC Configuration

```
#include <vtss_phy_10g_api.h>
```

Data Fields

- `BOOL enable`
- `vtss_phy_10g_tx_macro_t tx_macro`
- `vtss_phy_10g_rx_macro_t rx_macro`
- `u8 rx_ch`
- `u8 tx_ch`

7.133.1 Detailed Description

10G Phy Lane SYNC Configuration

Malibu Only

Definition at line 1130 of file vtss_phy_10g_api.h.

7.133.2 Field Documentation

7.133.2.1 enable

```
BOOL vtss_phy_10g_lane_sync_conf_t::enable
```

Enable/Disable LANE SYNC

Definition at line 1131 of file vtss_phy_10g_api.h.

7.133.2.2 tx_macro

```
vtss_phy_10g_tx_macro_t vtss_phy_10g_lane_sync_conf_t::tx_macro
```

Tx Macro to lane sync to (destination)

Definition at line 1132 of file vtss_phy_10g_api.h.

7.133.2.3 rx_macro

```
vtss_phy_10g_rx_macro_t vtss_phy_10g_lane_sync_conf_t::rx_macro
```

Rx Macro to lane sync from (Source)

Definition at line 1133 of file vtss_phy_10g_api.h.

7.133.2.4 rx_ch

```
u8 vtss_phy_10g_lane_sync_conf_t::rx_ch
```

0[Default] to 3- NA If rx_macro is SREFCLK

Definition at line 1134 of file vtss_phy_10g_api.h.

7.133.2.5 tx_ch

```
u8 vtss_phy_10g_lane_sync_conf_t::tx_ch
```

0[Default] to 3- NA If tx_macro is SCKOUT

Definition at line 1135 of file vtss_phy_10g_api.h.

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

- vtss_api/include/vtss_phy_10g_api.h

7.134 vtss_phy_10g_line_clk_conf_t Struct Reference

10G Phy Line clock config data

```
#include <vtss_phy_10g_api.h>
```

Data Fields

- vtss_phy_10g_clk_sel_t mode
- vtss_phy_10g_recvrd_clk_sel_t recvrd_clk_sel
- u8 clk_sel_no

7.134.1 Detailed Description

10G Phy Line clock config data

Malibu Only

Definition at line 1026 of file vtss_phy_10g_api.h.

7.134.2 Field Documentation

7.134.2.1 mode

`vtss_phy_10g_clk_sel_t` `vtss_phy_10g_line_clk_conf_t::mode`

Line side output clock mode

Definition at line 1027 of file vtss_phy_10g_api.h.

7.134.2.2 recvrd_clk_sel

`vtss_phy_10g_recvrd_clk_sel_t` `vtss_phy_10g_line_clk_conf_t::recvrd_clk_sel`

Recovered clock selection

Definition at line 1028 of file vtss_phy_10g_api.h.

7.134.2.3 clk_sel_no

`u8` `vtss_phy_10g_line_clk_conf_t::clk_sel_no`

Line clock select No(0-3)

Definition at line 1029 of file vtss_phy_10g_api.h.

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

- `vtss_api/include/vtss_phy_10g_api.h`

7.135 vtss_phy_10g_loopback_t Struct Reference

10G Phy system and network loopbacks

```
#include <vtss_phy_10g_api.h>
```

Data Fields

- `vtss_lb_type_t lb_type`
- `BOOL enable`

7.135.1 Detailed Description

10G Phy system and network loopbacks

Definition at line 1625 of file `vtss_phy_10g_api.h`.

7.135.2 Field Documentation

7.135.2.1 lb_type

`vtss_lb_type_t vtss_phy_10g_loopback_t::lb_type`

Looback types

Definition at line 1626 of file `vtss_phy_10g_api.h`.

7.135.2.2 enable

`BOOL vtss_phy_10g_loopback_t::enable`

Enable/Disable loopback given in <lb_type>

Definition at line 1627 of file `vtss_phy_10g_api.h`.

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

- `vtss_api/include/vtss_phy_10g_api.h`

7.136 vtss_phy_10g_mode_t Struct Reference

10G Phy operating mode

```
#include <vtss_phy_10g_api.h>
```

Public Types

- enum { `VTSS_EDC_FW_LOAD_MDIO`, `VTSS_EDC_FW_LOAD_NOTHING` }
EDC modes.

Data Fields

- `oper_mode_t oper_mode`
 - `vtss_phy_interface_mode interface`
 - `vtss_wrefclk_t wrefclk`
 - `BOOL high_input_gain`
 - `BOOL xfi_pol_invert`
 - `BOOL xaui_lane_flip`
 - `vtss_channel_t channel_id`
 - `BOOL hl_clk_synth`
 - `vtss_rcvrd_t rcvrd_clk`
 - `vtss_rcvrclk_cdr_div_t rcvrd_clk_div`
 - `vtss_srefclk_div_t sref_clk_div`
 - `vtss_wref_clk_div_t wref_clk_div`
 - `enum vtss_phy_10g_mode_t:: { ... } edc_fw_load`
- EDC modes.*
- `struct {`
 - `BOOL use_conf`
 - `BOOL ob_conf`
 - `BOOL ib_conf`
 - `BOOL dig_offset_reg`
 - `BOOL apc_offs_ctrl`
 - `BOOL apc_line_id_ctrl`
 - `BOOL apc_host_id_ctrl`
 - `u32 d_filter`
 - `u32 cfg0`
 - `u32 ib_ini_lp`
 - `u32 ib_min_lp`
 - `u32 ib_max_lp`
 - `u32 apc_eqz_offs_par_cfg`
 - `u32 apc_line_eqz_id_ctrl`
 - `u32 apc_host_eqz_id_ctrl`
 - `BOOL l_offset_guard`
 - `BOOL h_offset_guard`
- `} serdes_conf`

Serdes parameters.

- `apc_ib_regulator_t apc_ib_regulator`
- `u16 pma_txratecontrol`
- `BOOL venice_rev_a_los_detection_workaround`
- `ddr_mode_t ddr_mode`
- `clk_mstr_t master`
- `vtss_rptr_rate_t rate`
- `vtss_phy_10g_polarity_inv_t polarity`
- `BOOL is_host_wan`
- `vtss_phy_10g_clk_src_t h_clk_src`
- `vtss_phy_10g_clk_src_t l_clk_src`
- `BOOL lref_for_host`
- `vtss_phy_6g_link_partner_distance_t link_6g_distance`
- `vtss_phy_10g_media_t h_media`
- `vtss_phy_10g_media_t l_media`
- `vtss_phy_10g_ib_conf_t h_ib_conf`
- `vtss_phy_10g_ib_conf_t l_ib_conf`
- `vtss_phy_10g_apc_conf_t h_apc_conf`
- `vtss_phy_10g_apc_conf_t l_apc_conf`
- `BOOL enable_pass_thru`
- `BOOL is_init`
- `BOOL sd6g_calib_done`

7.136.1 Detailed Description

10G Phy operating mode

Definition at line 333 of file `vtss_phy_10g_api.h`.

7.136.2 Member Enumeration Documentation

7.136.2.1 anonymous enum

anonymous enum

EDC modes.

Enumerator

<code>VTSS_EDC_FW_LOAD_MDIO</code>	Load EDC FW through MDIO to iCPU
<code>VTSS_EDC_FW_LOAD_NOTHING</code>	Do not load FW to iCPU

Definition at line 357 of file `vtss_phy_10g_api.h`.

7.136.3 Field Documentation

7.136.3.1 oper_mode

`oper_mode_t vtss_phy_10g_mode_t::oper_mode`

Phy operational mode

Definition at line 334 of file `vtss_phy_10g_api.h`.

7.136.3.2 interface

`vtss_phy_interface_mode vtss_phy_10g_mode_t::interface`

Interface mode.

Definition at line 336 of file `vtss_phy_10g_api.h`.

7.136.3.3 wrefclk

`vtss_wrefclk_t` `vtss_phy_10g_mode_t::wrefclk`

848X only: WAN ref clock

Definition at line 338 of file vtss_phy_10g_api.h.

7.136.3.4 high_input_gain

`BOOL` `vtss_phy_10g_mode_t::high_input_gain`

Disable=0 (default), Enable=1. Should not be enabled unless needed

Definition at line 340 of file vtss_phy_10g_api.h.

7.136.3.5 xfi_pol_invert

`BOOL` `vtss_phy_10g_mode_t::xfi_pol_invert`

Selects polarity of the TX XFI data. 1:Invert 0:Normal

Definition at line 341 of file vtss_phy_10g_api.h.

7.136.3.6 xaui_lane_flip

`BOOL` `vtss_phy_10g_mode_t::xaui_lane_flip`

Swaps lane 0 <-> 3 and 1 <-> 2 for both RX and TX

Definition at line 342 of file vtss_phy_10g_api.h.

7.136.3.7 channel_id

`vtss_channel1_t` `vtss_phy_10g_mode_t::channel_id`

Channel id of this instance of the Phy

Definition at line 343 of file vtss_phy_10g_api.h.

7.136.3.8 hl_clk_synth

`BOOL vtss_phy_10g_mode_t::hl_clk_synth`

0: Free running clock 1: Hitless clock

Definition at line 346 of file vtss_phy_10g_api.h.

7.136.3.9 rcvrd_clk

`vtss_rcvrd_t vtss_phy_10g_mode_t::rcvrd_clk`

RXCLKOUT/TXCLKOUT used as recovered clock (not used any more, instead use the api functions: `vtss_phy_10g_rxckout_set` and `vtss_phy_10g_txckout_set`)

Definition at line 347 of file vtss_phy_10g_api.h.

7.136.3.10 rcvrd_clk_div

`vtss_rcvrdclk_cdr_div_t vtss_phy_10g_mode_t::rcvrd_clk_div`

8488 only: recovered clock's divisor

Definition at line 350 of file vtss_phy_10g_api.h.

7.136.3.11 sref_clk_div

`vtss_srefclk_div_t vtss_phy_10g_mode_t::sref_clk_div`

8488 only: SRERCLK divisor

Definition at line 351 of file vtss_phy_10g_api.h.

7.136.3.12 wref_clk_div

`vtss_wref_clk_div_t vtss_phy_10g_mode_t::wref_clk_div`

8488 only: WREFCLK divisor

Definition at line 352 of file vtss_phy_10g_api.h.

7.136.3.13 edc_fw_load

```
enum { ... } vtss_phy_10g_mode_t::edc_fw_load
```

EDC modes.

EDC Firmware load

7.136.3.14 use_conf

```
BOOL vtss_phy_10g_mode_t::use_conf
```

Use this configuration instead of default(only for setting 'd_filter'in Venice)

Definition at line 365 of file vtss_phy_10g_api.h.

7.136.3.15 ob_conf

```
BOOL vtss_phy_10g_mode_t::ob_conf
```

Configuration for SD10F OB instead of default (only for Venice family)

Definition at line 366 of file vtss_phy_10g_api.h.

7.136.3.16 ib_conf

```
BOOL vtss_phy_10g_mode_t::ib_conf
```

Configuration for SD6G ib_ini_lp, ib_min_lp & ib_max_lp (only for Venice family)

Definition at line 367 of file vtss_phy_10g_api.h.

7.136.3.17 dig_offset_reg

```
BOOL vtss_phy_10g_mode_t::dig_offset_reg
```

Digital offset regulation for SD6G IB. Default is Analog(only for Venice family)

Definition at line 368 of file vtss_phy_10g_api.h.

7.136.3.18 apc_offs_ctrl

`BOOL vtss_phy_10g_mode_t::apc_offs_ctrl`

Parameter used to control APC offset(overwrite APC_EQZ_OFFSET_PAR_CFG default value with apc_eqz_offset_par_cfg)

Definition at line 369 of file vtss_phy_10g_api.h.

7.136.3.19 apc_line_ld_ctrl

`BOOL vtss_phy_10g_mode_t::apc_line_ld_ctrl`

Parameter used to control APC Line LD Ctrl (overwrite LDLEV_INI, line apc value with apc_line_eqz_id_ctrl)

Definition at line 370 of file vtss_phy_10g_api.h.

7.136.3.20 apc_host_ld_ctrl

`BOOL vtss_phy_10g_mode_t::apc_host_ld_ctrl`

Parameter used to control APC Host LD Ctrl (overwrite LDLEV_INI, host apc value with apc_line_eqz_id_ctrl)

Definition at line 371 of file vtss_phy_10g_api.h.

7.136.3.21 d_filter

`u32 vtss_phy_10g_mode_t::d_filter`

SD10G Transmit filter coefficients for FIR taps (default 0x7DF820)

Definition at line 372 of file vtss_phy_10g_api.h.

7.136.3.22 cfg0

`u32 vtss_phy_10g_mode_t::cfg0`

SD10G OB CFG0 value, configurable by USER (only for Venice family)

Definition at line 373 of file vtss_phy_10g_api.h.

7.136.3.23 ib_ini_lp

`u32 vtss_phy_10g_mode_t::ib_ini_lp`

SD6G Init force value for low-pass gain regulation (default 1)

Definition at line 374 of file `vtss_phy_10g_api.h`.

7.136.3.24 ib_min_lp

`u32 vtss_phy_10g_mode_t::ib_min_lp`

SD6G Min value for low-pass gain regulation (default 0)

Definition at line 375 of file `vtss_phy_10g_api.h`.

7.136.3.25 ib_max_lp

`u32 vtss_phy_10g_mode_t::ib_max_lp`

SD6G Max value for low-pass gain regulation (default 63)

Definition at line 376 of file `vtss_phy_10g_api.h`.

7.136.3.26 apc_eqz_offs_par_cfg

`u32 vtss_phy_10g_mode_t::apc_eqz_offs_par_cfg`

APC EQZ_OFFSETS Parameter control(value of register APC_EQZ_OFFSETS_PAR_CFG,updated when `apc_offs_ctrl` is set)

Definition at line 377 of file `vtss_phy_10g_api.h`.

7.136.3.27 apc_line_eqz_ld_ctrl

`u32 vtss_phy_10g_mode_t::apc_line_eqz_ld_ctrl`

APC EQZ Line LD control(value of LDLEVINI, line apc value. Updated when `apc_line_ld_ctrl` is set) range = 0x18-0x2C default values (SR mode): Venice-A/B 0x18; Malibu-A, Venice-C 0x2C; Malibu-B 0x18

Definition at line 378 of file `vtss_phy_10g_api.h`.

7.136.3.28 apc_host_eqz_ld_ctrl

```
u32 vtss_phy_10g_mode_t::apc_host_eqz_ld_ctrl
```

APC EQZ Host LD control(value of LDLEV_INI, host apc value. Updated when apc_line_id_ctrl is set) range = 0x18-0x2C default values (SR mode): Venice-A/B 0x18; Malibu-A, Venice-C 0x2C; Malibu-B 0x18

Definition at line 380 of file vtss_phy_10g_api.h.

7.136.3.29 l_offset_guard

```
BOOL vtss_phy_10g_mode_t::l_offset_guard
```

This variable is deprecated, not to be used

Definition at line 382 of file vtss_phy_10g_api.h.

7.136.3.30 h_offset_guard

```
BOOL vtss_phy_10g_mode_t::h_offset_guard
```

This variable is deprecated, not to be used

Definition at line 383 of file vtss_phy_10g_api.h.

7.136.3.31 serdes_conf

```
struct { ... } vtss_phy_10g_mode_t::serdes_conf
```

Serdes parameters.

Serdes configuration

7.136.3.32 apc_ib_regulator

```
apc_ib_regulator_t vtss_phy_10g_mode_t::apc_ib_regulator
```

Analog Parameter Control / IB equalizer (only for Venice family)

Definition at line 386 of file vtss_phy_10g_api.h.

7.136.3.33 pma_txratecontrol

`u16 vtss_phy_10g_mode_t::pma_txratecontrol`

Normal pma_txratecontrol value to be restored when loopback is disabled

Definition at line 387 of file vtss_phy_10g_api.h.

7.136.3.34 venice_rev_a_los_detection_workaround

`BOOL vtss_phy_10g_mode_t::venice_rev_a_los_detection_workaround`

TRUE => LOS detection work around enabled. Requires interrupt handling

Definition at line 388 of file vtss_phy_10g_api.h.

7.136.3.35 ddr_mode

`ddr_mode_t vtss_phy_10g_mode_t::ddr_mode`

DDR Interleave mode

Definition at line 389 of file vtss_phy_10g_api.h.

7.136.3.36 master

`clk_mstr_t vtss_phy_10g_mode_t::master`

Clock Master

Definition at line 390 of file vtss_phy_10g_api.h.

7.136.3.37 rate

`vtss_rptr_rate_t vtss_phy_10g_mode_t::rate`

Data rate in repeater mode

Definition at line 391 of file vtss_phy_10g_api.h.

7.136.3.38 polarity

`vtss_phy_10g_polarity_inv_t vtss_phy_10g_mode_t::polarity`

polarity inversion configuration

Definition at line 392 of file vtss_phy_10g_api.h.

7.136.3.39 is_host_wan

`BOOL vtss_phy_10g_mode_t::is_host_wan`

Flag that gives information of WAN rate is supported at host interface

Definition at line 393 of file vtss_phy_10g_api.h.

7.136.3.40 h_clk_src

`vtss_phy_10g_clk_src_t vtss_phy_10g_mode_t::h_clk_src`

Host side clock configuration

Definition at line 394 of file vtss_phy_10g_api.h.

7.136.3.41 l_clk_src

`vtss_phy_10g_clk_src_t vtss_phy_10g_mode_t::l_clk_src`

Line side clock configuration

Definition at line 395 of file vtss_phy_10g_api.h.

7.136.3.42 lref_for_host

`BOOL vtss_phy_10g_mode_t::lref_for_host`

Clock source selection HREF or LREF on HOST side

Definition at line 396 of file vtss_phy_10g_api.h.

7.136.3.43 link_6g_distance

`vtss_phy_6g_link_partner_distance_t` `vtss_phy_10g_mode_t::link_6g_distance`

Gives information of link partner distance from 6G macro

Definition at line 397 of file vtss_phy_10g_api.h.

7.136.3.44 h_media

`vtss_phy_10g_media_t` `vtss_phy_10g_mode_t::h_media`

Gives information of media type connected on HOST direction

Definition at line 398 of file vtss_phy_10g_api.h.

7.136.3.45 l_media

`vtss_phy_10g_media_t` `vtss_phy_10g_mode_t::l_media`

Gives information of media type connected on LINE direction For Venice rev C and Malibu rev B, it is recommended to use smart control for the media type settings regardless what the actual media type application is used.

Definition at line 399 of file vtss_phy_10g_api.h.

7.136.3.46 h_ib_conf

`vtss_phy_10g_ib_conf_t` `vtss_phy_10g_mode_t::h_ib_conf`

Host Input buffer configuration

Definition at line 403 of file vtss_phy_10g_api.h.

7.136.3.47 l_ib_conf

`vtss_phy_10g_ib_conf_t` `vtss_phy_10g_mode_t::l_ib_conf`

Line Input buffer configuration

Definition at line 404 of file vtss_phy_10g_api.h.

7.136.3.48 h_apc_conf

`vtss_phy_10g_apc_conf_t vtss_phy_10g_mode_t::h_apc_conf`

HOST APC configuration

Definition at line 405 of file vtss_phy_10g_api.h.

7.136.3.49 l_apc_conf

`vtss_phy_10g_apc_conf_t vtss_phy_10g_mode_t::l_apc_conf`

LINE APC configuration

Definition at line 406 of file vtss_phy_10g_api.h.

7.136.3.50 enable_pass_thru

`BOOL vtss_phy_10g_mode_t::enable_pass_thru`

Enables Pass through mode in VENICE

Definition at line 407 of file vtss_phy_10g_api.h.

7.136.3.51 is_init

`BOOL vtss_phy_10g_mode_t::is_init`

To identify intialization Phase

Definition at line 408 of file vtss_phy_10g_api.h.

7.136.3.52 sd6g_calib_done

`BOOL vtss_phy_10g_mode_t::sd6g_calib_done`

to identify initialization Phase for ib calibration

Definition at line 409 of file vtss_phy_10g_api.h.

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

- vtss_api/include/[vtss_phy_10g_api.h](#)

7.137 vtss_phy_10g_ob_status_t Struct Reference

10G Phy OB status

```
#include <vtss_phy_10g_api.h>
```

Data Fields

- `u8 r_ctrl`
- `u8 c_ctrl`
- `u8 slew`
- `u8 levn`
- `u32 d_fltr`
- `int v3`
- `int vp`
- `int v4`
- `int v5`
- `BOOL is_host`

7.137.1 Detailed Description

10G Phy OB status

Definition at line 1171 of file vtss_phy_10g_api.h.

7.137.2 Field Documentation

7.137.2.1 r_ctrl

```
u8 vtss_phy_10g_ob_status_t::r_ctrl
```

slew rate r active value

Definition at line 1172 of file vtss_phy_10g_api.h.

7.137.2.2 c_ctrl

```
u8 vtss_phy_10g_ob_status_t::c_ctrl
```

slew rate c active value

Definition at line 1173 of file vtss_phy_10g_api.h.

7.137.2.3 slew

`u8 vtss_phy_10g_ob_status_t::slew`

slew rate

Definition at line 1174 of file vtss_phy_10g_api.h.

7.137.2.4 levn

`u8 vtss_phy_10g_ob_status_t::levn`

amplitude

Definition at line 1175 of file vtss_phy_10g_api.h.

7.137.2.5 d_fltr

`u32 vtss_phy_10g_ob_status_t::d_fltr`

d-filter value

Definition at line 1176 of file vtss_phy_10g_api.h.

7.137.2.6 v3

`int vtss_phy_10g_ob_status_t::v3`

d_filter tap v3

Definition at line 1177 of file vtss_phy_10g_api.h.

7.137.2.7 vp

`int vtss_phy_10g_ob_status_t::vp`

d_filter tap vp

Definition at line 1178 of file vtss_phy_10g_api.h.

7.137.2.8 v4

```
int vtss_phy_10g_ob_status_t::v4
```

d_filter tap v4

Definition at line 1179 of file vtss_phy_10g_api.h.

7.137.2.9 v5

```
int vtss_phy_10g_ob_status_t::v5
```

d_filter tap v5

Definition at line 1180 of file vtss_phy_10g_api.h.

7.137.2.10 is_host

```
BOOL vtss_phy_10g_ob_status_t::is_host
```

flag that says host/line

Definition at line 1181 of file vtss_phy_10g_api.h.

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

- vtss_api/include/[vtss_phy_10g_api.h](#)

7.138 vtss_phy_10g_pcs_prbs_gen_conf_t Struct Reference

```
#include <vtss_phy_10g_api.h>
```

Data Fields

- `BOOL prbs_gen`

7.138.1 Detailed Description

\ brief 10G PHY pcs prbs generator configuration

Definition at line 1941 of file vtss_phy_10g_api.h.

7.138.2 Field Documentation

7.138.2.1 prbs_gen

`BOOL vtss_phy_10g_pcs_prbs_gen_conf_t::prbs_gen`

enable or disable prbs test pattern mode on transmit path

Definition at line 1942 of file vtss_phy_10g_api.h.

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

- `vtss_api/include/vtss_phy_10g_api.h`

7.139 vtss_phy_10g_pcs_prbs_mon_conf_t Struct Reference

`#include <vtss_phy_10g_api.h>`

Data Fields

- `BOOL prbs_mon`
- `u32 error_counter`

7.139.1 Detailed Description

\ brief 10G PHY pcs prbs analyzer configuration

Definition at line 1976 of file vtss_phy_10g_api.h.

7.139.2 Field Documentation

7.139.2.1 prbs_mon

`BOOL vtss_phy_10g_pcs_prbs_mon_conf_t::prbs_mon`

enable or disable prbs test pattern mode on receive path

Definition at line 1977 of file vtss_phy_10g_api.h.

7.139.2.2 error_counter

```
u32 vtss_phy_10g_pcs_prbs_mon_conf_t::error_counter
```

Error counters for pcs prbs

Definition at line 1978 of file vtss_phy_10g_api.h.

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

- [vtss_api/include/vtss_phy_10g_api.h](#)

7.140 vtss_phy_10g_pkt_gen_conf_t Struct Reference

10G PHY Packet generator configuration

```
#include <vtss_phy_10g_api.h>
```

Data Fields

- [BOOL enable](#)
- [BOOL ptp](#)
- [BOOL ingress](#)
- [BOOL frames](#)
- [BOOL frame_single](#)
- [u16 etype](#)
- [u8 pkt_len](#)
- [u32 ipg_len](#)
- [vtss_mac_addr_t smac](#)
- [vtss_mac_addr_t dmac](#)
- [u8 ptp_ts_sec](#)
- [u8 ptp_ts_ns](#)
- [u8 srate](#)

7.140.1 Detailed Description

10G PHY Packet generator configuration

Definition at line 2133 of file vtss_phy_10g_api.h.

7.140.2 Field Documentation

7.140.2.1 enable

`BOOL vtss_phy_10g_pkt_gen_conf_t::enable`

Enable or disable packet generator

Definition at line 2134 of file vtss_phy_10g_api.h.

7.140.2.2 ptp

`BOOL vtss_phy_10g_pkt_gen_conf_t::ptp`

PTP or standard frame

Definition at line 2135 of file vtss_phy_10g_api.h.

7.140.2.3 ingress

`BOOL vtss_phy_10g_pkt_gen_conf_t::ingress`

Ingress or egress

Definition at line 2136 of file vtss_phy_10g_api.h.

7.140.2.4 frames

`BOOL vtss_phy_10g_pkt_gen_conf_t::frames`

frames or idles

Definition at line 2137 of file vtss_phy_10g_api.h.

7.140.2.5 frame_single

`BOOL vtss_phy_10g_pkt_gen_conf_t::frame_single`

Generate single packet

Definition at line 2138 of file vtss_phy_10g_api.h.

7.140.2.6 etype

`u16 vtss_phy_10g_pkt_gen_conf_t::etype`

Ethertype

Definition at line 2139 of file vtss_phy_10g_api.h.

7.140.2.7 pkt_len

`u8 vtss_phy_10g_pkt_gen_conf_t::pkt_len`

Packet length,min=64,max=16KB

Definition at line 2140 of file vtss_phy_10g_api.h.

7.140.2.8 ipg_len

`u32 vtss_phy_10g_pkt_gen_conf_t::ipg_len`

Inter Packet Gap

Definition at line 2141 of file vtss_phy_10g_api.h.

7.140.2.9 smac

`vtss_mac_addr_t vtss_phy_10g_pkt_gen_conf_t::smac`

Source MAC address

Definition at line 2142 of file vtss_phy_10g_api.h.

7.140.2.10 dmac

`vtss_mac_addr_t vtss_phy_10g_pkt_gen_conf_t::dmac`

Destination MAC address

Definition at line 2143 of file vtss_phy_10g_api.h.

7.140.2.11 ptp_ts_sec

```
u8 vtss_phy_10g_pkt_gen_conf_t::ptp_ts_sec
```

Seconds part of timestamp value

Definition at line 2144 of file vtss_phy_10g_api.h.

7.140.2.12 ptp_ts_ns

```
u8 vtss_phy_10g_pkt_gen_conf_t::ptp_ts_ns
```

NanoSeconds part of ts value

Definition at line 2145 of file vtss_phy_10g_api.h.

7.140.2.13 srate

```
u8 vtss_phy_10g_pkt_gen_conf_t::srate
```

Srate for ptp frames

Definition at line 2146 of file vtss_phy_10g_api.h.

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

- vtss_api/include/vtss_phy_10g_api.h

7.141 vtss_phy_10g_pkt_mon_conf_t Struct Reference

10G PHY Packet Monitor configuration

```
#include <vtss_phy_10g_api.h>
```

Data Fields

- [BOOL enable](#)
- [BOOL update](#)
- [vtss_phy_10g_pkt_mon_rst_t reset](#)
- [vtss_32_cntr_t good_crc](#)
- [vtss_32_cntr_t bad_crc](#)
- [vtss_32_cntr_t frag](#)
- [vtss_32_cntr_t lfault](#)
- [vtss_32_cntr_t ber](#)

7.141.1 Detailed Description

10G PHY Packet Monitor configuration

Definition at line 2178 of file vtss_phy_10g_api.h.

7.141.2 Field Documentation

7.141.2.1 enable

`BOOL vtss_phy_10g_pkt_mon_conf_t::enable`

Enable or disable packet monitor

Definition at line 2179 of file vtss_phy_10g_api.h.

7.141.2.2 update

`BOOL vtss_phy_10g_pkt_mon_conf_t::update`

update and reads monitor counters

Definition at line 2180 of file vtss_phy_10g_api.h.

7.141.2.3 reset

`vtss_phy_10g_pkt_mon_RST_t vtss_phy_10g_pkt_mon_conf_t::reset`

resets all monitor counters

Definition at line 2181 of file vtss_phy_10g_api.h.

7.141.2.4 good_crc

`vtss_32_cntr_t vtss_phy_10g_pkt_mon_conf_t::good_crc`

Good CRC packet count

Definition at line 2182 of file vtss_phy_10g_api.h.

7.141.2.5 bad_crc

```
vtss_32_cntr_t vtss_phy_10g_pkt_mon_conf_t::bad_crc
```

Bad CRC packet count

Definition at line 2183 of file vtss_phy_10g_api.h.

7.141.2.6 frag

```
vtss_32_cntr_t vtss_phy_10g_pkt_mon_conf_t::frag
```

Fragmented packet count

Definition at line 2184 of file vtss_phy_10g_api.h.

7.141.2.7 lfault

```
vtss_32_cntr_t vtss_phy_10g_pkt_mon_conf_t::lfault
```

Local fault packet count

Definition at line 2185 of file vtss_phy_10g_api.h.

7.141.2.8 ber

```
vtss_32_cntr_t vtss_phy_10g_pkt_mon_conf_t::ber
```

B-errored packet count

Definition at line 2186 of file vtss_phy_10g_api.h.

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

- vtss_api/include/[vtss_phy_10g_api.h](#)

7.142 vtss_phy_10g_polarity_inv_t Struct Reference

10G Phy Polarity inversion

```
#include <vtss_phy_10g_api.h>
```

Data Fields

- `BOOL line_rx`
- `BOOL line_tx`
- `BOOL host_rx`
- `BOOL host_tx`

7.142.1 Detailed Description

10G Phy Polarity inversion

Definition at line 162 of file `vtss_phy_10g_api.h`.

7.142.2 Field Documentation

7.142.2.1 `line_rx`

`BOOL vtss_phy_10g_polarity_inv_t::line_rx`

Line side Receive path

Definition at line 163 of file `vtss_phy_10g_api.h`.

7.142.2.2 `line_tx`

`BOOL vtss_phy_10g_polarity_inv_t::line_tx`

Line side Transmit path

Definition at line 164 of file `vtss_phy_10g_api.h`.

7.142.2.3 `host_rx`

`BOOL vtss_phy_10g_polarity_inv_t::host_rx`

Host side Receive path

Definition at line 165 of file `vtss_phy_10g_api.h`.

7.142.2.4 host_tx

`BOOL vtss_phy_10g_polarity_inv_t::host_tx`

Host side Transmit path

Definition at line 166 of file vtss_phy_10g_api.h.

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

- `vtss_api/include/vtss_phy_10g_api.h`

7.143 vtss_phy_10g_prbs_gen_conf_t Struct Reference

```
#include <vtss_phy_10g_api.h>
```

Data Fields

- `BOOL enable`
- `u8 prbsn_tx_sel`
- `BOOL line`
- `BOOL prbsn_tx_io`
- `u8 prbsn_tx_iw`

7.143.1 Detailed Description

\ brief 10G PHY prbs generator configuration

Definition at line 2026 of file vtss_phy_10g_api.h.

7.143.2 Field Documentation

7.143.2.1 enable

`BOOL vtss_phy_10g_prbs_gen_conf_t::enable`

enable or disable prbs generator

Definition at line 2027 of file vtss_phy_10g_api.h.

7.143.2.2 prbsn_tx_sel

`u8 vtss_phy_10g_prbs_gen_conf_t::prbsn_tx_sel`

select the prbs to be implemented, min=0, max=5

Definition at line 2028 of file `vtss_phy_10g_api.h`.

7.143.2.3 line

`BOOL vtss_phy_10g_prbs_gen_conf_t::line`

select the line side or host side, 1 for line side

Definition at line 2029 of file `vtss_phy_10g_api.h`.

7.143.2.4 prbsn_tx_io

`BOOL vtss_phy_10g_prbs_gen_conf_t::prbsn_tx_io`

Invert PRBS TX pattern

Definition at line 2030 of file `vtss_phy_10g_api.h`.

7.143.2.5 prbsn_tx_iw

`u8 vtss_phy_10g_prbs_gen_conf_t::prbsn_tx_iw`

select the prbs interface widtdh ,range 0-5

Definition at line 2031 of file `vtss_phy_10g_api.h`.

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

- `vtss_api/include/vtss_phy_10g_api.h`

7.144 vtss_phy_10g_prbs_mon_conf_t Struct Reference

10G PHY prbs monitor Configuration

```
#include <vtss_phy_10g_api.h>
```

Data Fields

- `BOOL enable`
- `BOOL line`
- `u16 max_bist_frames`
- `u16 error_states`
- `u16 des_interface_width`
- `u16 prbsn_sel`
- `BOOL prbs_check_input_invert`
- `u16 no_of_errors`
- `u16 bist_mode`
- `u32 error_status`
- `u32 PRBS_status`
- `u32 main_status`
- `BOOL stuck_at_par`
- `BOOL stuck_at_01`
- `BOOL no_sync`
- `BOOL instable`
- `BOOL incomplete`
- `BOOL active`

7.144.1 Detailed Description

10G PHY prbs monitor Configuration

Definition at line 2065 of file vtss_phy_10g_api.h.

7.144.2 Field Documentation

7.144.2.1 enable

`BOOL vtss_phy_10g_prbs_mon_conf_t::enable`

enable or disable the prbs monitor

Definition at line 2066 of file vtss_phy_10g_api.h.

7.144.2.2 line

`BOOL vtss_phy_10g_prbs_mon_conf_t::line`

select line side or host side, 1 for line side

Definition at line 2067 of file vtss_phy_10g_api.h.

7.144.2.3 max_bist_frames

`u16 vtss_phy_10g_prbs_mon_conf_t::max_bist_frames`

threshold to iterate counter for max_bist_frames [15:0]

Definition at line 2068 of file vtss_phy_10g_api.h.

7.144.2.4 error_states

`u16 vtss_phy_10g_prbs_mon_conf_t::error_states`

States in which error counting is enabled3:all but IDLE; 2:check 1:stable+check,0:wait_stable+stable+check

Definition at line 2069 of file vtss_phy_10g_api.h.

7.144.2.5 des_interface_width

`u16 vtss_phy_10g_prbs_mon_conf_t::des_interface_width`

DES interface width 0:8,1:10,2:16,3:20,4:32,5:40 (default)

Definition at line 2070 of file vtss_phy_10g_api.h.

7.144.2.6 prbsn_sel

`u16 vtss_phy_10g_prbs_mon_conf_t::prbsn_sel`

select the prbs to be implemented, min=0, max=5>

Definition at line 2071 of file vtss_phy_10g_api.h.

7.144.2.7 prbs_check_input_invert

`BOOL vtss_phy_10g_prbs_mon_conf_t::prbs_check_input_invert`

Enables PRBS checker input inversion

Definition at line 2072 of file vtss_phy_10g_api.h.

7.144.2.8 no_of_errors

`u16 vtss_phy_10g_prbs_mon_conf_t::no_of_errors`

Number of consecutive errors/non-errors before transitioning to respective state , value = num-40-bits-words + 1

Definition at line 2073 of file vtss_phy_10g_api.h.

7.144.2.9 bist_mode

`u16 vtss_phy_10g_prbs_mon_conf_t::bist_mode`

0: off, 1: BIST, 2: BER, 3:CONT(infinite mode)

Definition at line 2074 of file vtss_phy_10g_api.h.

7.144.2.10 error_status

`u32 vtss_phy_10g_prbs_mon_conf_t::error_status`

Error stautus of PRBS

Definition at line 2075 of file vtss_phy_10g_api.h.

7.144.2.11 PRBS_status

`u32 vtss_phy_10g_prbs_mon_conf_t::PRBS_status`

PRBS status

Definition at line 2076 of file vtss_phy_10g_api.h.

7.144.2.12 main_status

`u32 vtss_phy_10g_prbs_mon_conf_t::main_status`

Main stauts

Definition at line 2077 of file vtss_phy_10g_api.h.

7.144.2.13 stuck_at_par

`BOOL vtss_phy_10g_prbs_mon_conf_t::stuck_at_par`

Data input is unchanged for all 40 parallel bits for at least 7 clock cycles

Definition at line 2078 of file vtss_phy_10g_api.h.

7.144.2.14 stuck_at_01

`BOOL vtss_phy_10g_prbs_mon_conf_t::stuck_at_01`

Data input is unchanged for all 40 parallel bits for at least 7 clock cycles

Definition at line 2079 of file vtss_phy_10g_api.h.

7.144.2.15 no_sync

`BOOL vtss_phy_10g_prbs_mon_conf_t::no_sync`

no sync found since BIST enabled

Definition at line 2080 of file vtss_phy_10g_api.h.

7.144.2.16 instable

`BOOL vtss_phy_10g_prbs_mon_conf_t::instable`

BIST input data not stable

Definition at line 2081 of file vtss_phy_10g_api.h.

7.144.2.17 incomplete

`BOOL vtss_phy_10g_prbs_mon_conf_t::incomplete`

BIST not complete i.e. it has not reached a stable state

Definition at line 2082 of file vtss_phy_10g_api.h.

7.144.2.18 active

`BOOL vtss_phy_10g_prbs_mon_conf_t::active`

BIST is active but has not entered a final state

Definition at line 2083 of file vtss_phy_10g_api.h.

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

- `vtss_api/include/vtss_phy_10g_api.h`

7.145 vtss_phy_10g_rxckout_conf_t Struct Reference

10G Phy RXCKOUT config data

```
#include <vtss_phy_10g_api.h>
```

Data Fields

- `vtss_recvrd_clkout_t mode`
- `BOOL squelch_on_pcs_fault`
- `BOOL squelch_on_lopc`

7.145.1 Detailed Description

10G Phy RXCKOUT config data

Definition at line 706 of file vtss_phy_10g_api.h.

7.145.2 Field Documentation

7.145.2.1 mode

`vtss_recvrd_clkout_t vtss_phy_10g_rxckout_conf_t::mode`

RXCKOUT output mode (DISABLE/RX_CLK/TX_CLK)

Definition at line 707 of file vtss_phy_10g_api.h.

7.145.2.2 squelch_on_pcs_fault

`BOOL vtss_phy_10g_rxckout_conf_t::squelch_on_pcs_fault`

Enable squelching on PCS_FAULT (supported on revision no = 1 (Rev C) and above)

Definition at line 708 of file `vtss_phy_10g_api.h`.

7.145.2.3 squelch_on_lopc

`BOOL vtss_phy_10g_rxckout_conf_t::squelch_on_lopc`

Enable squelching on LOPC (supported on revision no = 1 (Rev C) and above)

Definition at line 709 of file `vtss_phy_10g_api.h`.

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

- `vtss_api/include/vtss_phy_10g_api.h`

7.146 vtss_phy_10g_sckout_conf_t Struct Reference

10G Phy SCKOUT config data

```
#include <vtss_phy_10g_api.h>
```

Data Fields

- `vtss_phy_10g_clk_sel_t mode`
- `vtss_phy_10g_squelch_src_t src`
- `vtss_phy_10g_sckout_freq_t freq`
- `BOOL squelch_inv`
- `BOOL enable`

7.146.1 Detailed Description

10G Phy SCKOUT config data

Malibu Only

Definition at line 982 of file `vtss_phy_10g_api.h`.

7.146.2 Field Documentation

7.146.2.1 mode

`vtss_phy_10g_clk_sel_t vtss_phy_10g_sckout_conf_t::mode`

SCKOUT output clock mode

Definition at line 983 of file vtss_phy_10g_api.h.

7.146.2.2 src

`vtss_phy_10g_squelch_src_t vtss_phy_10g_sckout_conf_t::src`

SCKOUT squelch source

Definition at line 984 of file vtss_phy_10g_api.h.

7.146.2.3 freq

`vtss_phy_10g_sckout_freq_t vtss_phy_10g_sckout_conf_t::freq`

SCKOUT freq(156.25MHz, 125MHz only)

Definition at line 985 of file vtss_phy_10g_api.h.

7.146.2.4 squelch_inv

`BOOL vtss_phy_10g_sckout_conf_t::squelch_inv`

'0'- Use squelch source src as is, '1'-Invert

Definition at line 986 of file vtss_phy_10g_api.h.

7.146.2.5 enable

`BOOL vtss_phy_10g_sckout_conf_t::enable`

Enable/Disable SCKOUT

Definition at line 987 of file vtss_phy_10g_api.h.

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

- vtss_api/include/[vtss_phy_10g_api.h](#)

7.147 vtss_phy_10g_serdes_status_t Struct Reference

10G Phy SERDES status

```
#include <vtss_phy_10g_api.h>
```

Data Fields

- u8 rcomp
- BOOL h_pll5g_lock_status
- BOOL h_pll5g_fsm_lock
- u8 h_pll5g_fsm_stat
- u8 h_pll5g_gain
- BOOL l_pll5g_lock_status
- BOOL l_pll5g_fsm_lock
- u8 l_pll5g_fsm_stat
- u8 l_pll5g_gain
- BOOL h_rx_rcpll_lock_status
- u8 h_rx_rcpll_range
- u8 h_rx_rcpll_vco_load
- u8 h_rx_rcpll_fsm_status
- BOOL l_rx_rcpll_lock_status
- u8 l_rx_rcpll_range
- u8 l_rx_rcpll_vco_load
- u8 l_rx_rcpll_fsm_status
- BOOL h_tx_rcpll_lock_status
- u8 h_tx_rcpll_range
- u8 h_tx_rcpll_vco_load
- u8 h_tx_rcpll_fsm_status
- BOOL l_tx_rcpll_lock_status
- u8 l_tx_rcpll_range
- u8 l_tx_rcpll_vco_load
- u8 l_tx_rcpll_fsm_status
- vtss_sublayer_status_t h_pma
- vtss_sublayer_status_t h_pcs
- vtss_sublayer_status_t l_pma
- vtss_sublayer_status_t l_pcs
- vtss_sublayer_status_t wis

7.147.1 Detailed Description

10G Phy SERDES status

Definition at line 253 of file vtss_phy_10g_api.h.

7.147.2 Field Documentation

7.147.2.1 rcomp

`u8 vtss_phy_10g_serdes_status_t::rcomp`

Measured Resistor value

Definition at line 254 of file vtss_phy_10g_api.h.

7.147.2.2 h_pll5g_lock_status

`BOOL vtss_phy_10g_serdes_status_t::h_pll5g_lock_status`

TRUE value says its locked

Definition at line 257 of file vtss_phy_10g_api.h.

7.147.2.3 h_pll5g_fsm_lock

`BOOL vtss_phy_10g_serdes_status_t::h_pll5g_fsm_lock`

TRUE value says fsm is locked

Definition at line 258 of file vtss_phy_10g_api.h.

7.147.2.4 h_pll5g_fsm_stat

`u8 vtss_phy_10g_serdes_status_t::h_pll5g_fsm_stat`

FSM status

Definition at line 259 of file vtss_phy_10g_api.h.

7.147.2.5 h_pll5g_gain

`u8 vtss_phy_10g_serdes_status_t::h_pll5g_gain`

Gain

Definition at line 260 of file vtss_phy_10g_api.h.

7.147.2.6 l_pll5g_lock_status

`BOOL vtss_phy_10g_serdes_status_t::l_pll5g_lock_status`

TRUE value says its locked

Definition at line 263 of file vtss_phy_10g_api.h.

7.147.2.7 l_pll5g_fsm_lock

`BOOL vtss_phy_10g_serdes_status_t::l_pll5g_fsm_lock`

TRUE value says fsm is locked

Definition at line 264 of file vtss_phy_10g_api.h.

7.147.2.8 l_pll5g_fsm_stat

`u8 vtss_phy_10g_serdes_status_t::l_pll5g_fsm_stat`

FSM status

Definition at line 265 of file vtss_phy_10g_api.h.

7.147.2.9 l_pll5g_gain

`u8 vtss_phy_10g_serdes_status_t::l_pll5g_gain`

Gain

Definition at line 266 of file vtss_phy_10g_api.h.

7.147.2.10 h_rx_rcpll_lock_status

`BOOL vtss_phy_10g_serdes_status_t::h_rx_rcpll_lock_status`

TRUE value says its locked

Definition at line 269 of file vtss_phy_10g_api.h.

7.147.2.11 h_rx_rcpll_range

`u8 vtss_phy_10g_serdes_status_t::h_rx_rcpll_range`

TRUE value says with in range

Definition at line 270 of file vtss_phy_10g_api.h.

7.147.2.12 h_rx_rcpll_vco_load

`u8 vtss_phy_10g_serdes_status_t::h_rx_rcpll_vco_load`

Actual value of VCV load

Definition at line 271 of file vtss_phy_10g_api.h.

7.147.2.13 h_rx_rcpll_fsm_status

`u8 vtss_phy_10g_serdes_status_t::h_rx_rcpll_fsm_status`

Actual value of FSM stage

Definition at line 272 of file vtss_phy_10g_api.h.

7.147.2.14 l_rx_rcpll_lock_status

`BOOL vtss_phy_10g_serdes_status_t::l_rx_rcpll_lock_status`

TRUE value says its locked

Definition at line 273 of file vtss_phy_10g_api.h.

7.147.2.15 l_rx_rcpll_range

`u8 vtss_phy_10g_serdes_status_t::l_rx_rcpll_range`

TRUE value says with in range

Definition at line 274 of file vtss_phy_10g_api.h.

7.147.2.16 l_rx_rcpll_vco_load

`u8 vtss_phy_10g_serdes_status_t::l_rx_rcpll_vco_load`

Actual value of VCV load

Definition at line 275 of file `vtss_phy_10g_api.h`.

7.147.2.17 l_rx_rcpll_fsm_status

`u8 vtss_phy_10g_serdes_status_t::l_rx_rcpll_fsm_status`

Actual value of FSM stage

Definition at line 276 of file `vtss_phy_10g_api.h`.

7.147.2.18 h_tx_rcpll_lock_status

`BOOL vtss_phy_10g_serdes_status_t::h_tx_rcpll_lock_status`

TRUE value says its locked

Definition at line 279 of file `vtss_phy_10g_api.h`.

7.147.2.19 h_tx_rcpll_range

`u8 vtss_phy_10g_serdes_status_t::h_tx_rcpll_range`

TRUE value says with in range

Definition at line 280 of file `vtss_phy_10g_api.h`.

7.147.2.20 h_tx_rcpll_vco_load

`u8 vtss_phy_10g_serdes_status_t::h_tx_rcpll_vco_load`

Actual value of VCV load

Definition at line 281 of file `vtss_phy_10g_api.h`.

7.147.2.21 h_tx_rcpll_fsm_status

```
u8 vtss_phy_10g_serdes_status_t::h_tx_rcpll_fsm_status
```

Actual value of FSM stage

Definition at line 282 of file vtss_phy_10g_api.h.

7.147.2.22 l_tx_rcpll_lock_status

```
BOOL vtss_phy_10g_serdes_status_t::l_tx_rcpll_lock_status
```

TRUE value says its locked

Definition at line 283 of file vtss_phy_10g_api.h.

7.147.2.23 l_tx_rcpll_range

```
u8 vtss_phy_10g_serdes_status_t::l_tx_rcpll_range
```

TRUE value says with in range

Definition at line 284 of file vtss_phy_10g_api.h.

7.147.2.24 l_tx_rcpll_vco_load

```
u8 vtss_phy_10g_serdes_status_t::l_tx_rcpll_vco_load
```

Actual value of VCV load

Definition at line 285 of file vtss_phy_10g_api.h.

7.147.2.25 l_tx_rcpll_fsm_status

```
u8 vtss_phy_10g_serdes_status_t::l_tx_rcpll_fsm_status
```

Actual value of FSM stage

Definition at line 286 of file vtss_phy_10g_api.h.

7.147.2.26 h_pma

`vtss_sublayer_status_t vtss_phy_10g_serdes_status_t::h_pma`

Host pma status

Definition at line 289 of file `vtss_phy_10g_api.h`.

7.147.2.27 h_pcs

`vtss_sublayer_status_t vtss_phy_10g_serdes_status_t::h_pcs`

Host pcs status

Definition at line 290 of file `vtss_phy_10g_api.h`.

7.147.2.28 l_pma

`vtss_sublayer_status_t vtss_phy_10g_serdes_status_t::l_pma`

Line pma status

Definition at line 293 of file `vtss_phy_10g_api.h`.

7.147.2.29 l_pcs

`vtss_sublayer_status_t vtss_phy_10g_serdes_status_t::l_pcs`

Line pcs status

Definition at line 294 of file `vtss_phy_10g_api.h`.

7.147.2.30 wis

`vtss_sublayer_status_t vtss_phy_10g_serdes_status_t::wis`

WIS status

Definition at line 297 of file `vtss_phy_10g_api.h`.

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

- `vtss_api/include/vtss_phy_10g_api.h`

7.148 vtss_phy_10g_srefclk_mode_t Struct Reference

10G Phy srefclk config data

```
#include <vtss_phy_10g_api.h>
```

Data Fields

- `BOOL enable`
- `vtss_phy_10g_srefclk_freq_t freq`

7.148.1 Detailed Description

10G Phy srefclk config data

Definition at line 787 of file vtss_phy_10g_api.h.

7.148.2 Field Documentation

7.148.2.1 enable

```
BOOL vtss_phy_10g_srefclk_mode_t::enable
```

Enable locking line tx clock to srefclk input

Definition at line 788 of file vtss_phy_10g_api.h.

7.148.2.2 freq

```
vtss_phy_10g_srefclk_freq_t vtss_phy_10g_srefclk_mode_t::freq
```

The srefclk input frequency

Definition at line 789 of file vtss_phy_10g_api.h.

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

- vtss_api/include/[vtss_phy_10g_api.h](#)

7.149 vtss_phy_10g_status_t Struct Reference

10G Phy link and fault status for all sublayers

```
#include <vtss_phy_10g_api.h>
```

Data Fields

- `vtss_sublayer_status_t pma`
- `vtss_sublayer_status_t hpma`
- `vtss_sublayer_status_t wis`
- `vtss_sublayer_status_t pcs`
- `vtss_sublayer_status_t hpcs`
- `vtss_sublayer_status_t xs`
- `BOOL lpcs_1g`
- `BOOL hpcs_1g`
- `BOOL status`
- `BOOL block_lock`
- `BOOL lopc_stat`

7.149.1 Detailed Description

10G Phy link and fault status for all sublayers

Definition at line 1428 of file `vtss_phy_10g_api.h`.

7.149.2 Field Documentation

7.149.2.1 pma

`vtss_sublayer_status_t vtss_phy_10g_status_t::pma`

Status for Line PMA sublayer

Definition at line 1429 of file `vtss_phy_10g_api.h`.

7.149.2.2 hpma

`vtss_sublayer_status_t vtss_phy_10g_status_t::hpma`

Status for Host PMA sublayer

Definition at line 1430 of file `vtss_phy_10g_api.h`.

7.149.2.3 wis

`vtss_sublayer_status_t vtss_phy_10g_status_t::wis`

Status for WIS sublayer

Definition at line 1431 of file `vtss_phy_10g_api.h`.

7.149.2.4 pcs

`vtss_sublayer_status_t vtss_phy_10g_status_t::pcs`

Status for Line PCS sublayer

Definition at line 1432 of file vtss_phy_10g_api.h.

7.149.2.5 hpcs

`vtss_sublayer_status_t vtss_phy_10g_status_t::hpcs`

Status for HOST PCS sublayer,pcs xau in case of venice

Definition at line 1433 of file vtss_phy_10g_api.h.

7.149.2.6 xs

`vtss_sublayer_status_t vtss_phy_10g_status_t::xs`

Status for XAUI sublayer

Definition at line 1434 of file vtss_phy_10g_api.h.

7.149.2.7 lpcs_1g

`BOOL vtss_phy_10g_status_t::lpcs_1g`

Status for Line 1G_PCS sublayer

Definition at line 1435 of file vtss_phy_10g_api.h.

7.149.2.8 hpcs_1g

`BOOL vtss_phy_10g_status_t::hpcs_1g`

Status for Host 1G_PCS sublayer

Definition at line 1436 of file vtss_phy_10g_api.h.

7.149.2.9 status

`BOOL vtss_phy_10g_status_t::status`

Status of whole PHY , based on operation mode and PHY type

Definition at line 1437 of file vtss_phy_10g_api.h.

7.149.2.10 block_lock

`BOOL vtss_phy_10g_status_t::block_lock`

Gives block lock information

Definition at line 1438 of file vtss_phy_10g_api.h.

7.149.2.11 lopc_stat

`BOOL vtss_phy_10g_status_t::lopc_stat`

LOPC status

Definition at line 1439 of file vtss_phy_10g_api.h.

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

- `vtss_api/include/vtss_phy_10g_api.h`

7.150 vtss_phy_10g_timestamp_val_t Struct Reference

10G PHY timestamp value array(holder)

```
#include <vtss_phy_10g_api.h>
```

Data Fields

- `u16 timestamp [10][5]`

7.150.1 Detailed Description

10G PHY timestamp value array(holder)

Definition at line 2190 of file vtss_phy_10g_api.h.

7.150.2 Field Documentation

7.150.2.1 timestamp

`u16 vtss_phy_10g_timestamp_val_t::timestamp[10][5]`

5 bytes each of 10 timestamp values

Definition at line 2191 of file `vtss_phy_10g_api.h`.

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

- `vtss_api/include/vtss_phy_10g_api.h`

7.151 `vtss_phy_10g_txckout_conf_t` Struct Reference

10G Phy TXCKOUT config data

```
#include <vtss_phy_10g_api.h>
```

Data Fields

- `vtss_recvrd_clkout_t mode`

7.151.1 Detailed Description

10G Phy TXCKOUT config data

Definition at line 743 of file `vtss_phy_10g_api.h`.

7.151.2 Field Documentation

7.151.2.1 mode

`vtss_recvrd_clkout_t vtss_phy_10g_txckout_conf_t::mode`

TXCKOUT output mode (DISABLE/RX_CLK/TX_CLK)

Definition at line 744 of file `vtss_phy_10g_api.h`.

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

- `vtss_api/include/vtss_phy_10g_api.h`

7.152 vtss_phy_10g_vscope_conf_t Struct Reference

```
#include <vtss_phy_10g_api.h>
```

Data Fields

- `vtss_phy_10g_vscope_scan_t scan_type`
- `BOOL line`
- `BOOL enable`
- `u32 error_thres`

7.152.1 Detailed Description

\ brief VSCOPE scan configuration

Definition at line 1856 of file vtss_phy_10g_api.h.

7.152.2 Field Documentation

7.152.2.1 scan_type

```
vtss_phy_10g_vscope_scan_t vtss_phy_10g_vscope_conf_t::scan_type
```

selects the type of scan to be implemented

Definition at line 1857 of file vtss_phy_10g_api.h.

7.152.2.2 line

```
BOOL vtss_phy_10g_vscope_conf_t::line
```

select line side or host side, TRUE for line side

Definition at line 1858 of file vtss_phy_10g_api.h.

7.152.2.3 enable

```
BOOL vtss_phy_10g_vscope_conf_t::enable
```

enable or disable vscope fast scan

Definition at line 1859 of file vtss_phy_10g_api.h.

7.152.2.4 error_thres

`u32 vtss_phy_10g_vscope_conf_t::error_thres`

error_threshold for vscope calculations

Definition at line 1860 of file vtss_phy_10g_api.h.

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

- `vtss_api/include/vtss_phy_10g_api.h`

7.153 vtss_phy_10g_vscope_scan_conf_t Struct Reference

VSCOPE scan configuration.

```
#include <vtss_phy_10g_api.h>
```

Data Fields

- `BOOL line`
- `u32 x_start`
- `u32 y_start`
- `u32 x_incr`
- `u32 y_incr`
- `u32 x_count`
- `u32 y_count`
- `u32 ber`

7.153.1 Detailed Description

VSCOPE scan configuration.

Definition at line 1904 of file vtss_phy_10g_api.h.

7.153.2 Field Documentation

7.153.2.1 line

`BOOL vtss_phy_10g_vscope_scan_conf_t::line`

selects line or host side, 1 for line

Definition at line 1905 of file vtss_phy_10g_api.h.

7.153.2.2 x_start

`u32 vtss_phy_10g_vscope_scan_conf_t::x_start`

start value for x (0-127)

Definition at line 1906 of file vtss_phy_10g_api.h.

7.153.2.3 y_start

`u32 vtss_phy_10g_vscope_scan_conf_t::y_start`

start value for y (0-63)

Definition at line 1907 of file vtss_phy_10g_api.h.

7.153.2.4 x_incr

`u32 vtss_phy_10g_vscope_scan_conf_t::x_incr`

increment value for x during the scan

Definition at line 1908 of file vtss_phy_10g_api.h.

7.153.2.5 y_incr

`u32 vtss_phy_10g_vscope_scan_conf_t::y_incr`

increment value for y during the scan

Definition at line 1909 of file vtss_phy_10g_api.h.

7.153.2.6 x_count

`u32 vtss_phy_10g_vscope_scan_conf_t::x_count`

max value for x (upto which scan is to be performed)

Definition at line 1910 of file vtss_phy_10g_api.h.

7.153.2.7 y_count

`u32 vtss_phy_10g_vscope_scan_conf_t::y_count`

max value for y (upto which scan is to be performed)

Definition at line 1911 of file vtss_phy_10g_api.h.

7.153.2.8 ber

`u32 vtss_phy_10g_vscope_scan_conf_t::ber`

bit error rate

Definition at line 1912 of file vtss_phy_10g_api.h.

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

- vtss_api/include/vtss_phy_10g_api.h

7.154 vtss_phy_10g_vscope_scan_status_t Struct Reference

#include <vtss_phy_10g_api.h>

Data Fields

- [vtss_phy_10g_vscope_scan_conf_t scan_conf](#)
- [i32 error_free_x](#)
- [i32 error_free_y](#)
- [i32 amp_range](#)
- [u32 errors \[PHASE_POINTS\]\[AMPLITUDE_POINTS\]](#)

7.154.1 Detailed Description

\ brief Vsphere eye scan status

Definition at line 1919 of file vtss_phy_10g_api.h.

7.154.2 Field Documentation

7.154.2.1 scan_conf

`vtss_phy_10g_vscope_scan_conf_t vtss_phy_10g_vscope_scan_status_t::scan_conf`

scan configuration data

Definition at line 1920 of file `vtss_phy_10g_api.h`.

7.154.2.2 error_free_x

`i32 vtss_phy_10g_vscope_scan_status_t::error_free_x`

error free x values in case of fast eye scan

Definition at line 1921 of file `vtss_phy_10g_api.h`.

7.154.2.3 error_free_y

`i32 vtss_phy_10g_vscope_scan_status_t::error_free_y`

error free y values in case of fast eye scan

Definition at line 1922 of file `vtss_phy_10g_api.h`.

7.154.2.4 amp_range

`i32 vtss_phy_10g_vscope_scan_status_t::amp_range`

amp range in case of fast eye scan

Definition at line 1923 of file `vtss_phy_10g_api.h`.

7.154.2.5 errors

`u32 vtss_phy_10g_vscope_scan_status_t::errors [PHASE_POINTS] [AMPLITUDE_POINTS]`

error matrix in full scan mode

Definition at line 1924 of file `vtss_phy_10g_api.h`.

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

- `vtss_api/include/vtss_phy_10g_api.h`

7.155 vtss_phy_aneg_t Struct Reference

PHY auto negotiation advertisement.

```
#include <vtss_phy_api.h>
```

Data Fields

- `BOOL speed_10m_hdx`
- `BOOL speed_10m_fdx`
- `BOOL speed_100m_hdx`
- `BOOL speed_100m_fdx`
- `BOOL speed_1g_fdx`
- `BOOL speed_1g_hdx`
- `BOOL symmetric_pause`
- `BOOL asymmetric_pause`
- `BOOL tx_remote_fault`

7.155.1 Detailed Description

PHY auto negotiation advertisement.

Definition at line 309 of file vtss_phy_api.h.

7.155.2 Field Documentation

7.155.2.1 speed_10m_hdx

```
BOOL vtss_phy_aneg_t::speed_10m_hdx
```

10Mbps, half duplex

Definition at line 310 of file vtss_phy_api.h.

7.155.2.2 speed_10m_fdx

```
BOOL vtss_phy_aneg_t::speed_10m_fdx
```

10Mbps, full duplex

Definition at line 311 of file vtss_phy_api.h.

7.155.2.3 speed_100m_hdx

`BOOL vtss_phy_aneg_t::speed_100m_hdx`

100Mbps, half duplex

Definition at line 312 of file `vtss_phy_api.h`.

7.155.2.4 speed_100m_fdx

`BOOL vtss_phy_aneg_t::speed_100m_fdx`

100Mbps, full duplex

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

7.155.2.5 speed_1g_fdx

`BOOL vtss_phy_aneg_t::speed_1g_fdx`

1000Mbps, full duplex

Definition at line 314 of file `vtss_phy_api.h`.

7.155.2.6 speed_1g_hdx

`BOOL vtss_phy_aneg_t::speed_1g_hdx`

1000Mbps, full duplex

Definition at line 315 of file `vtss_phy_api.h`.

7.155.2.7 symmetric_pause

`BOOL vtss_phy_aneg_t::symmetric_pause`

Symmetric pause

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

7.155.2.8 asymmetric_pause

`BOOL vtss_phy_aneg_t::asymmetric_pause`

Asymmetric pause

Definition at line 317 of file vtss_phy_api.h.

7.155.2.9 tx_remote_fault

`BOOL vtss_phy_aneg_t::tx_remote_fault`

Local Application fault indication for Link Partner

Definition at line 318 of file vtss_phy_api.h.

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

- vtss_api/include/[vtss_phy_api.h](#)

7.156 vtss_phy_clock_conf_t Struct Reference

PHY clock configuration.

```
#include <vtss_phy_api.h>
```

Data Fields

- [vtss_phy_clk_source_t src](#)
- [vtss_phy_freq_t freq](#)
- [vtss_phy_clk_squelch squelch](#)

7.156.1 Detailed Description

PHY clock configuration.

Definition at line 648 of file vtss_phy_api.h.

7.156.2 Field Documentation

7.156.2.1 src

`vtss_phy_clk_source_t` `vtss_phy_clock_conf_t::src`

Clock source

Definition at line 649 of file `vtss_phy_api.h`.

7.156.2.2 freq

`vtss_phy_freq_t` `vtss_phy_clock_conf_t::freq`

Clock frequency

Definition at line 650 of file `vtss_phy_api.h`.

7.156.2.3 squelch

`vtss_phy_clk_squelch` `vtss_phy_clock_conf_t::squelch`

Clock squelch level

Definition at line 651 of file `vtss_phy_api.h`.

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

- `vtss_api/include/vtss_phy_api.h`

7.157 vtss_phy_conf_1g_t Struct Reference

PHY 1G configuration.

```
#include <vtss_phy_api.h>
```

Data Fields

- struct {
 BOOL cfg
 BOOL val
} master

7.157.1 Detailed Description

PHY 1G configuration.

Definition at line 504 of file vtss_phy_api.h.

7.157.2 Field Documentation

7.157.2.1 cfg

`BOOL vtss_phy_conf_1g_t::cfg`

Manual Master/Slave Config. 1=enabled

Definition at line 506 of file vtss_phy_api.h.

7.157.2.2 val

`BOOL vtss_phy_conf_1g_t::val`

Master/Slave Config value, 1=Master

Definition at line 507 of file vtss_phy_api.h.

7.157.2.3 master

`struct { ... } vtss_phy_conf_1g_t::master`

Master/Slave Mode

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

- vtss_api/include/[vtss_phy_api.h](#)

7.158 vtss_phy_conf_t Struct Reference

PHY configuration.

```
#include <vtss_phy_api.h>
```

Data Fields

- `vtss_phy_mode_t mode`
- `vtss_phy_forced_t forced`
- `vtss_phy_aneg_t aneg`
- `vtss_phy_mdi_t mdi`
- `vtss_phy_fast_link_fail_t flf`
- `vtss_phy_sigdet_polarity_t sigdet`
- `vtss_phy_unidirectional_t unidir`
- `vtss_phy_mac_serdes_pcs_ctrl_t mac_if_pcs`
- `vtss_phy_media_serdes_pcs_ctrl_t media_if_pcs`
- `vtss_phy_media_force_ams_sel_t force_ams_sel`

7.158.1 Detailed Description

PHY configuration.

Definition at line 396 of file `vtss_phy_api.h`.

7.158.2 Field Documentation

7.158.2.1 mode

`vtss_phy_mode_t vtss_phy_conf_t::mode`

PHY mode

Definition at line 397 of file `vtss_phy_api.h`.

7.158.2.2 forced

`vtss_phy_forced_t vtss_phy_conf_t::forced`

Forced mode configuration

Definition at line 398 of file `vtss_phy_api.h`.

7.158.2.3 aneg

`vtss_phy_aneg_t vtss_phy_conf_t::aneg`

Auto-negotiation mode configuration

Definition at line 399 of file `vtss_phy_api.h`.

7.158.2.4 mdi

`vtss_phy_mdi_t` `vtss_phy_conf_t::mdi`

Cu cable MDI (Crossed cable / normal cable)

Definition at line 400 of file vtss_phy_api.h.

7.158.2.5 flf

`vtss_phy_fast_link_fail_t` `vtss_phy_conf_t::flf`

Fast link failure configuration

Definition at line 401 of file vtss_phy_api.h.

7.158.2.6 sigdet

`vtss_phy_sigdet_polarity_t` `vtss_phy_conf_t::sigdet`

Sigdet pin polarity configuration

Definition at line 402 of file vtss_phy_api.h.

7.158.2.7 unidir

`vtss_phy_unidirectional_t` `vtss_phy_conf_t::unidir`

Unidirectional Configuration

Definition at line 403 of file vtss_phy_api.h.

7.158.2.8 mac_if_pcs

`vtss_phy_mac_serdes_pcs_ctrl_t` `vtss_phy_conf_t::mac_if_pcs`

PHY MAC SerDes PCS Control (Reg16E3)

Definition at line 404 of file vtss_phy_api.h.

7.158.2.9 media_if_pcs

`vtss_phy_media_serdes_pcs_cntl_t` `vtss_phy_conf_t::media_if_pcs`

PHY MAC SerDes PCS Control (Reg23E3)

Definition at line 405 of file `vtss_phy_api.h`.

7.158.2.10 force_ams_sel

`vtss_phy_media_force_ams_sel_t` `vtss_phy_conf_t::force_ams_sel`

PHY Media AMS Force Selection

Definition at line 406 of file `vtss_phy_api.h`.

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

- `vtss_api/include/vtss_phy_api.h`

7.159 vtss_phy_daisy_chain_conf_t Struct Reference

SPI daisy chain configuration.

```
#include <vtss_phy_ts_api.h>
```

Data Fields

- `BOOL spi_daisy_input`
- `BOOL spi_daisy_output`

7.159.1 Detailed Description

SPI daisy chain configuration.

Definition at line 535 of file `vtss_phy_ts_api.h`.

7.159.2 Field Documentation

7.159.2.1 spi_daisy_input

`BOOL vtss_phy_daisy_chain_conf_t::spi_daisy_input`

Enable SPI daisy-chain input port

Definition at line 536 of file `vtss_phy_ts_api.h`.

7.159.2.2 spi_daisy_output

`BOOL vtss_phy_daisy_chain_conf_t::spi_daisy_output`

Enable SPI daisy-chain output port

Definition at line 537 of file `vtss_phy_ts_api.h`.

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

- `vtss_api/include/vtss_phy_ts_api.h`

7.160 vtss_phy_eee_conf_t Struct Reference

EEE configuration.

```
#include <vtss_phy_api.h>
```

Data Fields

- `vtss_eee_mode_t eee_mode`
- `BOOL eee_ena_phy`

7.160.1 Detailed Description

EEE configuration.

Definition at line 987 of file `vtss_phy_api.h`.

7.160.2 Field Documentation

7.160.2.1 eee_mode

`vtss_eee_mode_t vtss_phy_eee_conf_t::eee_mode`

EEE mode.

Definition at line 988 of file `vtss_phy_api.h`.

7.160.2.2 eee_ena_phy

`BOOL vtss_phy_eee_conf_t::eee_ena_phy`

Signaling current state in the phy api

Definition at line 989 of file `vtss_phy_api.h`.

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

- `vtss_api/include/vtss_phy_api.h`

7.161 vtss_phy_enhanced_led_control_t Struct Reference

enhanced LED control

`#include <vtss_phy_api.h>`

Data Fields

- `BOOL ser_led_output_1`
- `BOOL ser_led_output_2`
- `u8 ser_led_frame_rate`
- `u8 ser_led_select`

7.161.1 Detailed Description

enhanced LED control

Definition at line 1010 of file `vtss_phy_api.h`.

7.161.2 Field Documentation

7.161.2.1 ser_led_output_1

```
BOOL vtss_phy_enhanced_led_control_t::ser_led_output_1
```

Set to TRUE if hardware board uses serial LEDs at PHY0, PHY1, PHY2 and PHY3 LED0 pins.

Definition at line 1011 of file vtss_phy_api.h.

7.161.2.2 ser_led_output_2

```
BOOL vtss_phy_enhanced_led_control_t::ser_led_output_2
```

Set to TRUE if hardware board uses serial LEDs at PHY0, PHY1, PHY2 and PHY3 LED1 pins.

Definition at line 1012 of file vtss_phy_api.h.

7.161.2.3 ser_led_frame_rate

```
u8 vtss_phy_enhanced_led_control_t::ser_led_frame_rate
```

Serial LED frame rate. 0x0 = 2500Hz, 0x1 = 1000 Hz, 0x2 = 500 Hz, 0x3 = 250 Hz, 0x4 = 200 Hz, 0x5 = 125 Hz, 0x6 = 40 Hz

Definition at line 1013 of file vtss_phy_api.h.

7.161.2.4 ser_led_select

```
u8 vtss_phy_enhanced_led_control_t::ser_led_select
```

The number of LEDs the hardware board supports for each PHY, 0x00 = 4 LEDs, 0x01 = 3 LEDs, 0x02 = 2 LEDs, 0x03 = 1 LED

Definition at line 1014 of file vtss_phy_api.h.

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

- vtss_api/include/[vtss_phy_api.h](#)

7.162 vtss_phy_forced_t Struct Reference

PHY forced mode configuration.

```
#include <vtss_phy_api.h>
```

Data Fields

- `vtss_port_speed_t speed`
- `BOOL fdx`

7.162.1 Detailed Description

PHY forced mode configuration.

Definition at line 303 of file `vtss_phy_api.h`.

7.162.2 Field Documentation

7.162.2.1 speed

`vtss_port_speed_t vtss_phy_forced_t::speed`

Speed

Definition at line 304 of file `vtss_phy_api.h`.

7.162.2.2 fdx

`BOOL vtss_phy_forced_t::fdx`

Full duplex

Definition at line 305 of file `vtss_phy_api.h`.

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

- `vtss_api/include/vtss_phy_api.h`

7.163 vtss_phy_led_mode_select_t Struct Reference

LED model selection.

```
#include <vtss_phy_api.h>
```

Data Fields

- `vtss_phy_led_mode_t mode`
- `vtss_phy_led_number_t number`

7.163.1 Detailed Description

LED model selection.

Definition at line 136 of file vtss_phy_api.h.

7.163.2 Field Documentation

7.163.2.1 mode

`vtss_phy_led_mode_t vtss_phy_led_mode_select_t::mode`

LED blink mode

Definition at line 138 of file vtss_phy_api.h.

7.163.2.2 number

`vtss_phy_led_number_t vtss_phy_led_mode_select_t::number`

Which LED to configure with the above mode

Definition at line 139 of file vtss_phy_api.h.

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

- `vtss_api/include/vtss_phy_api.h`

7.164 vtss_phy_loopback_t Struct Reference

1G Phy loopbacks

```
#include <vtss_phy_api.h>
```

Data Fields

- `BOOL far_end_enable`
- `BOOL near_end_enable`
- `BOOL connector_enable`
- `BOOL mac_serdes_input_enable`
- `BOOL mac_serdes_facility_enable`
- `BOOL mac_serdes_equipment_enable`
- `BOOL media_serdes_input_enable`
- `BOOL media_serdes_facility_enable`
- `BOOL media_serdes_equipment_enable`

7.164.1 Detailed Description

1G Phy loopbacks

Definition at line 1384 of file vtss_phy_api.h.

7.164.2 Field Documentation

7.164.2.1 far_end_enable

```
BOOL vtss_phy_loopback_t::far_end_enable
```

Enable/Disable loopback far end loopback

Definition at line 1385 of file vtss_phy_api.h.

7.164.2.2 near_end_enable

```
BOOL vtss_phy_loopback_t::near_end_enable
```

Enable/Disable loopback near end loopback

Definition at line 1386 of file vtss_phy_api.h.

7.164.2.3 connector_enable

```
BOOL vtss_phy_loopback_t::connector_enable
```

Enable/Disable loopback connector loopback

Definition at line 1387 of file vtss_phy_api.h.

7.164.2.4 mac_serdes_input_enable

```
BOOL vtss_phy_loopback_t::mac_serdes_input_enable
```

Enable/Disable loopback SerDes MAC Input loopback

Definition at line 1388 of file vtss_phy_api.h.

7.164.2.5 mac_serdes_facility_enable

```
BOOL vtss_phy_loopback_t::mac_serdes_facility_enable
```

Enable/Disable loopback SerDes MAC Facility loopback

Definition at line 1389 of file vtss_phy_api.h.

7.164.2.6 mac_serdes_equipment_enable

```
BOOL vtss_phy_loopback_t::mac_serdes_equipment_enable
```

Enable/Disable loopback SerDes MAC Equipment loopback

Definition at line 1390 of file vtss_phy_api.h.

7.164.2.7 media_serdes_input_enable

```
BOOL vtss_phy_loopback_t::media_serdes_input_enable
```

Enable/Disable loopback SerDes MEDIA Input loopback

Definition at line 1391 of file vtss_phy_api.h.

7.164.2.8 media_serdes_facility_enable

```
BOOL vtss_phy_loopback_t::media_serdes_facility_enable
```

Enable/Disable loopback SerDes MEDIA Facility loopback

Definition at line 1392 of file vtss_phy_api.h.

7.164.2.9 media_serdes_equipment_enable

```
BOOL vtss_phy_loopback_t::media_serdes_equipment_enable
```

Enable/Disable loopback SerDes MEDIA Equipment loopback

Definition at line 1393 of file vtss_phy_api.h.

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

- vtss_api/include/[vtss_phy_api.h](#)

7.165 vtss_phy_ltc_freq_synth_s Struct Reference

Frequency synthesis pulse configuration.

```
#include <vtss_phy_ts_api.h>
```

Data Fields

- `BOOL enable`
- `u8 high_duty_cycle`
- `u8 low_duty_cycle`

7.165.1 Detailed Description

Frequency synthesis pulse configuration.

Definition at line 501 of file vtss_phy_ts_api.h.

7.165.2 Field Documentation

7.165.2.1 enable

```
BOOL vtss_phy_ltc_freq_synth_s::enable
```

Enable/Disable frequency synthesis pulse

Definition at line 502 of file vtss_phy_ts_api.h.

7.165.2.2 high_duty_cycle

```
u8 vtss_phy_ltc_freq_synth_s::high_duty_cycle
```

Number of clock cycles pulse is high

Definition at line 503 of file vtss_phy_ts_api.h.

7.165.2.3 low_duty_cycle

u8 vtss_phy_ltc_freq_synth_s::low_duty_cycle

Number of clock cycles pulse is low

Definition at line 504 of file vtss_phy_ts_api.h.

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

- vtss_api/include/vtss_phy_ts_api.h

7.166 vtss_phy_mac_serdes_pcs_ctrl_t Struct Reference

PHY MAC SerDes PCS Control, Reg16E3.

```
#include <vtss_phy_api.h>
```

Data Fields

- BOOL disable
- BOOL restart
- BOOL pd_enable
- BOOL aneg_restart
- BOOL force_adv_ability
- vtss_phy_mac_serdes_pcs_sgmii_pre sgmii_in_pre
- BOOL sgmii_out_pre
- BOOL serdes_aneg_ena
- BOOL serdes_pol_inv_in
- BOOL serdes_pol_inv_out
- BOOL fast_link_stat_ena
- BOOL inhibit_odd_start

7.166.1 Detailed Description

PHY MAC SerDes PCS Control, Reg16E3.

Definition at line 351 of file vtss_phy_api.h.

7.166.2 Field Documentation

7.166.2.1 disable

`BOOL vtss_phy_mac_serdes_pcs_ctrl_t::disable`

MAC i/f disable: 1000BaseX MAC i/f disable when media link down

Definition at line 352 of file `vtss_phy_api.h`.

7.166.2.2 restart

`BOOL vtss_phy_mac_serdes_pcs_ctrl_t::restart`

MAC i/f restart: 1000BaseX MAC i/f restart on media link change

Definition at line 353 of file `vtss_phy_api.h`.

7.166.2.3 pd_enable

`BOOL vtss_phy_mac_serdes_pcs_ctrl_t::pd_enable`

MAC i/f ANEG parallel detect enable

Definition at line 354 of file `vtss_phy_api.h`.

7.166.2.4 aneg_restart

`BOOL vtss_phy_mac_serdes_pcs_ctrl_t::aneg_restart`

Restart MAC i/f ANEG

Definition at line 355 of file `vtss_phy_api.h`.

7.166.2.5 force_adv_ability

`BOOL vtss_phy_mac_serdes_pcs_ctrl_t::force_adv_ability`

Force adv. ability from Reg18E3

Definition at line 356 of file `vtss_phy_api.h`.

7.166.2.6 sgmii_in_pre

```
vtss_phy_mac_serdes_pcs_sgmii_pre vtss_phy_mac_serdes_pcs_ctrl_t::sgmii_in_pre
```

SGMII Input Preamble for 100BaseFX

Definition at line 357 of file vtss_phy_api.h.

7.166.2.7 sgmii_out_pre

```
BOOL vtss_phy_mac_serdes_pcs_ctrl_t::sgmii_out_pre
```

SGMII Output Preamble

Definition at line 358 of file vtss_phy_api.h.

7.166.2.8 serdes_aneg_ena

```
BOOL vtss_phy_mac_serdes_pcs_ctrl_t::serdes_aneg_ena
```

MAC SerDes ANEG Enable

Definition at line 359 of file vtss_phy_api.h.

7.166.2.9 serdes_pol_inv_in

```
BOOL vtss_phy_mac_serdes_pcs_ctrl_t::serdes_pol_inv_in
```

Invert SerDes Polarity at input of MAC

Definition at line 360 of file vtss_phy_api.h.

7.166.2.10 serdes_pol_inv_out

```
BOOL vtss_phy_mac_serdes_pcs_ctrl_t::serdes_pol_inv_out
```

Invert SerDes Polarity at output of MAC

Definition at line 361 of file vtss_phy_api.h.

7.166.2.11 fast_link_stat_ena

`BOOL vtss_phy_mac_serdes_pcs_cntl_t::fast_link_stat_ena`

Fast Link Fail Status Enable

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

7.166.2.12 inhibit_odd_start

`BOOL vtss_phy_mac_serdes_pcs_cntl_t::inhibit_odd_start`

Inhibit MAC Odd-Start delay

Definition at line 363 of file `vtss_phy_api.h`.

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

- `vtss_api/include/vtss_phy_api.h`

7.167 `vtss_phy_media_serdes_pcs_cntl_t` Struct Reference

PHY MEDIA SerDes PCS Control, Reg23E3.

```
#include <vtss_phy_api.h>
```

Data Fields

- `vtss_phy_media_rem_fault_t remote_fault`
- `BOOL aneg_pd_detect`
- `BOOL force_adv_ability`
- `BOOL serdes_pol_inv_in`
- `BOOL serdes_pol_inv_out`
- `BOOL inhibit_odd_start`
- `BOOL force_hls`
- `BOOL force_fefi`
- `BOOL force_fefi_value`

7.167.1 Detailed Description

PHY MEDIA SerDes PCS Control, Reg23E3.

Definition at line 375 of file `vtss_phy_api.h`.

7.167.2 Field Documentation

7.167.2.1 remote_fault

```
vtss_phy_media_rem_fault_t vtss_phy_media_serd_pcs_ctrl_t::remote_fault
```

Remote Fault to Media indication sent in most recent Clause 37 ANEG

Definition at line 376 of file vtss_phy_api.h.

7.167.2.2 aneg_pd_detect

```
BOOL vtss_phy_media_serd_pcs_ctrl_t::aneg_pd_detect
```

SerDes MEDIA ANEG parallel detect enable

Definition at line 377 of file vtss_phy_api.h.

7.167.2.3 force_adv_ability

```
BOOL vtss_phy_media_serd_pcs_ctrl_t::force_adv_ability
```

Force adv. ability from Reg25E3

Definition at line 378 of file vtss_phy_api.h.

7.167.2.4 serdes_pol_inv_in

```
BOOL vtss_phy_media_serd_pcs_ctrl_t::serdes_pol_inv_in
```

Invert SerDes Polarity at input of Media SerDes

Definition at line 379 of file vtss_phy_api.h.

7.167.2.5 serdes_pol_inv_out

```
BOOL vtss_phy_media_serd_pcs_ctrl_t::serdes_pol_inv_out
```

Invert SerDes Polarity at output of Media SerDes

Definition at line 380 of file vtss_phy_api.h.

7.167.2.6 inhibit_odd_start

`BOOL vtss_phy_media_serdes_pcs_ctrl_t::inhibit_odd_start`

Inhibit Media Odd-Start delay

Definition at line 381 of file `vtss_phy_api.h`.

7.167.2.7 force_hls

`BOOL vtss_phy_media_serdes_pcs_ctrl_t::force_hls`

Forces 100BaseFX to Tx HSL continuously

Definition at line 382 of file `vtss_phy_api.h`.

7.167.2.8 force_fefi

`BOOL vtss_phy_media_serdes_pcs_ctrl_t::force_fefi`

Forces 100BaseFX Far-End-Fault Indication

Definition at line 383 of file `vtss_phy_api.h`.

7.167.2.9 force_fefi_value

`BOOL vtss_phy_media_serdes_pcs_ctrl_t::force_fefi_value`

Forces/Suppress FEFI

Definition at line 384 of file `vtss_phy_api.h`.

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

- `vtss_api/include/vtss_phy_api.h`

7.168 vtss_phy_pcs_cnt_t Struct Reference

10G Phy PCS counters

```
#include <vtss_phy_10g_api.h>
```

Data Fields

- `BOOL block_lock_latched`
- `BOOL high_ber_latched`
- `u8 ber_cnt`
- `u8 err_blk_cnt`

7.168.1 Detailed Description

10G Phy PCS counters

Definition at line 1668 of file `vtss_phy_10g_api.h`.

7.168.2 Field Documentation

7.168.2.1 `block_lock_latched`

`BOOL vtss_phy_pcs_cnt_t::block_lock_latched`

Latched block status

Definition at line 1669 of file `vtss_phy_10g_api.h`.

7.168.2.2 `high_ber_latched`

`BOOL vtss_phy_pcs_cnt_t::high_ber_latched`

Latched high ber status

Definition at line 1670 of file `vtss_phy_10g_api.h`.

7.168.2.3 `ber_cnt`

`u8 vtss_phy_pcs_cnt_t::ber_cnt`

BER counter. Saturating, clear on read

Definition at line 1671 of file `vtss_phy_10g_api.h`.

7.168.2.4 err_blk_cnt

```
u8 vtss_phy_pcs_cnt_t::err_blk_cnt
```

ERROR block counter. Saturating, clear on read

Definition at line 1672 of file vtss_phy_10g_api.h.

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

- [vtss_api/include/vtss_phy_10g_api.h](#)

7.169 vtss_phy_power_conf_t Struct Reference

PHY power configuration.

```
#include <vtss_phy_api.h>
```

Data Fields

- [vtss_phy_power_mode_t mode](#)

7.169.1 Detailed Description

PHY power configuration.

Definition at line 562 of file vtss_phy_api.h.

7.169.2 Field Documentation

7.169.2.1 mode

```
vtss_phy_power_mode_t vtss_phy_power_conf_t::mode
```

Power mode

Definition at line 563 of file vtss_phy_api.h.

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

- [vtss_api/include/vtss_phy_api.h](#)

7.170 vtss_phy_power_status_t Struct Reference

PHY power status.

```
#include <vtss_phy_api.h>
```

Data Fields

- [u32 level](#)

7.170.1 Detailed Description

PHY power status.

Definition at line 597 of file vtss_phy_api.h.

7.170.2 Field Documentation

7.170.2.1 level

```
u32 vtss_phy_power_status_t::level
```

Usage level

Definition at line 598 of file vtss_phy_api.h.

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

- vtss_api/include/[vtss_phy_api.h](#)

7.171 vtss_phy_reset_conf_t Struct Reference

PHY reset structure.

```
#include <vtss_phy_api.h>
```

Data Fields

- [vtss_port_interface_t mac_if](#)
- [vtss_phy_media_interface_t media_if](#)
- [vtss_phy_rgmii_conf_t rgmii](#)
- [vtss_phy_tbi_conf_t tbi](#)
- [vtss_phy_forced_reset_t force](#)
- [vtss_phy_pkt_mode_t pkt_mode](#)
- [BOOL i_cpu_en](#)

7.171.1 Detailed Description

PHY reset structure.

Definition at line 218 of file vtss_phy_api.h.

7.171.2 Field Documentation

7.171.2.1 mac_if

`vtss_port_interface_t` vtss_phy_reset_conf_t::mac_if

MAC interface

Definition at line 219 of file vtss_phy_api.h.

7.171.2.2 media_if

`vtss_phy_media_interface_t` vtss_phy_reset_conf_t::media_if

Media interface

Definition at line 220 of file vtss_phy_api.h.

7.171.2.3 rgmii

`vtss_phy_rgmii_conf_t` vtss_phy_reset_conf_t::rgmii

RGMII MAC interface setup

Definition at line 221 of file vtss_phy_api.h.

7.171.2.4 tbi

`vtss_phy_tbi_conf_t` vtss_phy_reset_conf_t::tbi

TBI setup

Definition at line 222 of file vtss_phy_api.h.

7.171.2.5 force

```
vtss_phy_forced_reset_t vtss_phy_reset_conf_t::force
```

Force or NoForce PHY port Reset during vtss_phy_reset_private, Only used for Selected PHY Families

Definition at line 223 of file vtss_phy_api.h.

7.171.2.6 pkt_mode

```
vtss_phy_pkt_mode_t vtss_phy_reset_conf_t::pkt_mode
```

packet mode

Definition at line 224 of file vtss_phy_api.h.

7.171.2.7 i_cpu_en

```
BOOL vtss_phy_reset_conf_t::i_cpu_en
```

Set to TRUE to enable internal 8051 CPU (Enzo and Spyder family only)

Definition at line 225 of file vtss_phy_api.h.

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

- vtss_api/include/vtss_phy_api.h

7.172 vtss_phy_rgmii_conf_t Struct Reference

PHY RGMII configuration.

```
#include <vtss_phy_api.h>
```

Data Fields

- u16 rx_clk_skew_ps
- u16 tx_clk_skew_ps

7.172.1 Detailed Description

PHY RGMII configuration.

Definition at line 194 of file vtss_phy_api.h.

7.172.2 Field Documentation

7.172.2.1 rx_clk_skew_ps

```
u16 vtss_phy_rgmii_conf_t::rx_clk_skew_ps
```

Rx clock skew in pico seconds, see rgmii_skew_delay_psec_t for options

Definition at line 195 of file vtss_phy_api.h.

7.172.2.2 tx_clk_skew_ps

```
u16 vtss_phy_rgmii_conf_t::tx_clk_skew_ps
```

Tx clock skew in pico seconds, see rgmii_skew_delay_psec_t for options

Definition at line 196 of file vtss_phy_api.h.

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

- vtss_api/include/[vtss_phy_api.h](#)

7.173 vtss_phy_statistic_t Struct Reference

Phy statistic information.

```
#include <vtss_phy_api.h>
```

Data Fields

- u8 cu_good
- u8 cu_bad
- u16 serdes_tx_good
- u8 serdes_tx_bad
- u8 rx_err_cnt_base_tx
- u16 media_mac_serdes_good
- u8 media_mac_serdes_crc

7.173.1 Detailed Description

Phy statistic information.

Definition at line 1348 of file vtss_phy_api.h.

7.173.2 Field Documentation

7.173.2.1 cu_good

```
u8 vtss_phy_statistic_t::cu_good
```

Cu media CRC good packet received since last time read

Definition at line 1350 of file vtss_phy_api.h.

7.173.2.2 cu_bad

```
u8 vtss_phy_statistic_t::cu_bad
```

RC error counter for packets received on the Cu media interface. The value saturates at 0xFF and subsequently clears when read and restarts count.

Definition at line 1351 of file vtss_phy_api.h.

7.173.2.3 serdes_tx_good

```
u16 vtss_phy_statistic_t::serdes_tx_good
```

SerDes Transmit good packet count modulo 10000

Definition at line 1352 of file vtss_phy_api.h.

7.173.2.4 serdes_tx_bad

```
u8 vtss_phy_statistic_t::serdes_tx_bad
```

SerDes Transmit CRC packet count (saturates at 255)

Definition at line 1353 of file vtss_phy_api.h.

7.173.2.5 rx_err_cnt_base_tx

`u8 vtss_phy_statistic_t::rx_err_cnt_base_tx`

100/1000BASE-TX receive error counter. 8-bit counter that saturates when it reaches

1. These bits are self-clearing when read.

Definition at line 1354 of file `vtss_phy_api.h`.

7.173.2.6 media_mac_serdes_good

`u16 vtss_phy_statistic_t::media_mac_serdes_good`

Counter containing the number of packets with valid CRCs. This counter does not saturate and will roll over to 0 when the count reaches 10,000 packets.

Definition at line 1356 of file `vtss_phy_api.h`.

7.173.2.7 media_mac_serdes_crc

`u8 vtss_phy_statistic_t::media_mac_serdes_crc`

CRC error counter for packets received on the Fiber media or MAC interfaces. The value saturates at 0xFF and subsequently clears when read and restarts count.

Definition at line 1357 of file `vtss_phy_api.h`.

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

- `vtss_api/include/vtss_phy_api.h`

7.174 vtss_phy_status_1g_t Struct Reference

PHY 1G status.

```
#include <vtss_phy_api.h>
```

Data Fields

- `BOOL master_cfg_fault`
- `BOOL master`

7.174.1 Detailed Description

PHY 1G status.

Definition at line 538 of file vtss_phy_api.h.

7.174.2 Field Documentation

7.174.2.1 master_cfg_fault

`BOOL vtss_phy_status_1g_t::master_cfg_fault`

Master/Slave Configuration fault

Definition at line 539 of file vtss_phy_api.h.

7.174.2.2 master

`BOOL vtss_phy_status_1g_t::master`

Master = 1, Slave = 0

Definition at line 540 of file vtss_phy_api.h.

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

- `vtss_api/include/vtss_phy_api.h`

7.175 vtss_phy_tbi_conf_t Struct Reference

PHY TBI configuration.

```
#include <vtss_phy_api.h>
```

Data Fields

- `BOOL aneg_enable`

7.175.1 Detailed Description

PHY TBI configuration.

Definition at line 200 of file vtss_phy_api.h.

7.175.2 Field Documentation

7.175.2.1 aneg_enable

`BOOL vtss_phy_tbi_conf_t::aneg_enable`

Enable auto negotiation

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

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

- `vtss_api/include/vtss_phy_api.h`

7.176 `vtss_phy_timestamp_t` Struct Reference

PHY timestamp in seconds and nanoseconds (10 bytes Timestamp)

```
#include <vtss_phy_ts_api.h>
```

Data Fields

- `struct {
 u16 high
 u32 low
} seconds`
- `u32 nanoseconds`

7.176.1 Detailed Description

PHY timestamp in seconds and nanoseconds (10 bytes Timestamp)

Definition at line 250 of file `vtss_phy_ts_api.h`.

7.176.2 Field Documentation

7.176.2.1 high

`u16 vtss_phy_timestamp_t::high`

bits 32-47 of 48-bit second

Definition at line 252 of file `vtss_phy_ts_api.h`.

7.176.2.2 low

`u32 vtss_phy_timestamp_t::low`

bits 0-31 of 48-bit second

Definition at line 253 of file vtss_phy_ts_api.h.

7.176.2.3 seconds

`struct { ... } vtss_phy_timestamp_t::seconds`

6 bytes second part of Timestamp

7.176.2.4 nanoseconds

`u32 vtss_phy_timestamp_t::nanoseconds`

4 bytes nano-sec part of Timestamp

Definition at line 255 of file vtss_phy_ts_api.h.

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

- vtss_api/include/vtss_phy_ts_api.h

7.177 vtss_phy_ts_ach_conf_t Struct Reference

Analyzer ACH comparator configuration options.

```
#include <vtss_phy_ts_api.h>
```

Data Fields

- `struct {
 struct {
 u8 value
 u8 mask
 } version
 struct {
 u16 value
 u16 mask
 } channel_type
 struct {
 u16 value
 u16 mask
 } proto_id
} comm_opt`

7.177.1 Detailed Description

Analyzer ACH comparator configuration options.

Note

ACH uses the IP1 comparator for match. So IP1 and ACH can not be used at the same time.

Definition at line 1099 of file vtss_phy_ts_api.h.

7.177.2 Field Documentation

7.177.2.1 value [1/2]

```
u8 vtss_phy_ts_ach_conf_t::value
```

4-bits version

Definition at line 1102 of file vtss_phy_ts_api.h.

7.177.2.2 mask [1/2]

```
u8 vtss_phy_ts_ach_conf_t::mask
```

Mask

Definition at line 1103 of file vtss_phy_ts_api.h.

7.177.2.3 version

```
struct { ... } vtss_phy_ts_ach_conf_t::version
```

version number of the PWACH in value/mask; set mask 0 for don't care

7.177.2.4 value [2/2]

```
u16 vtss_phy_ts_ach_conf_t::value
```

Channel type value

Protocol Identifier Value

Definition at line 1106 of file vtss_phy_ts_api.h.

7.177.2.5 mask [2/2]

`u16 vtss_phy_ts_ach_conf_t::mask`

Channel type mask

Protocol Identifier Mask

Definition at line 1107 of file vtss_phy_ts_api.h.

7.177.2.6 channel_type

`struct { ... } vtss_phy_ts_ach_conf_t::channel_type`

PW Associated Channel Type in value/mask format

7.177.2.7 proto_id

`struct { ... } vtss_phy_ts_ach_conf_t::proto_id`

PID: identifier of payload as defined in RFC 5718, only for PTP

7.177.2.8 comm_opt

`struct { ... } vtss_phy_ts_ach_conf_t::comm_opt`

ACH common config

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

- `vtss_api/include/vtss_phy_ts_api.h`

7.178 vtss_phy_ts_alt_clock_mode_s Struct Reference

parameter for setting the alternative clock mode.

```
#include <vtss_phy_ts_api.h>
```

Data Fields

- `BOOL pps_ls_lpbk`
- `BOOL ls_lpbk`
- `BOOL ls_pps_lpbk`

7.178.1 Detailed Description

parameter for setting the alternative clock mode.

external clock output configuration.

Definition at line 52 of file vtss_phy_ts_api.h.

7.178.2 Field Documentation

7.178.2.1 pps_ls_lpbk

`BOOL vtss_phy_ts_alt_clock_mode_s::pps_ls_lpbk`

output PPS is loopback to L/S input pin

Definition at line 53 of file vtss_phy_ts_api.h.

7.178.2.2 ls_lpbk

`BOOL vtss_phy_ts_alt_clock_mode_s::ls_lpbk`

L/S act as output pin at 1PPS

Definition at line 54 of file vtss_phy_ts_api.h.

7.178.2.3 ls_pps_lpbk

`BOOL vtss_phy_ts_alt_clock_mode_s::ls_pps_lpbk`

L/S connected to PPS out

Definition at line 55 of file vtss_phy_ts_api.h.

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

- vtss_api/include/[vtss_phy_ts_api.h](#)

7.179 vtss_phy_ts_eng_init_conf_t Struct Reference

Defines the basic engine parameters passed to the engine_init_conf() function.

```
#include <vtss_phy_ts_api.h>
```

Data Fields

- `BOOL eng_used`
- `vtss_phy_ts_encap_t encapsulation`
- `vtss_phy_ts_engine_flow_match_t flow_match_mode`
- `u8 flow_st_index`
- `u8 flow_end_index`

7.179.1 Detailed Description

Defines the basic engine parameters passed to the `engine_init_conf_get()` function.

Definition at line 848 of file `vtss_phy_ts_api.h`.

7.179.2 Field Documentation

7.179.2.1 eng_used

`BOOL vtss_phy_ts_eng_init_conf_t::eng_used`

allocated the engine to application

Definition at line 849 of file `vtss_phy_ts_api.h`.

7.179.2.2 encapsulation

`vtss_phy_ts_encap_t vtss_phy_ts_eng_init_conf_t::encapsulation`

engine encapsulation

Definition at line 850 of file `vtss_phy_ts_api.h`.

7.179.2.3 flow_match_mode

`vtss_phy_ts_engine_flow_match_t vtss_phy_ts_eng_init_conf_t::flow_match_mode`

strict/non-strict flow match

Definition at line 851 of file `vtss_phy_ts_api.h`.

7.179.2.4 flow_st_index

`u8 vtss_phy_ts_eng_init_conf_t::flow_st_index`

start index of flow

Definition at line 852 of file `vtss_phy_ts_api.h`.

7.179.2.5 flow_end_index

`u8 vtss_phy_ts_eng_init_conf_t::flow_end_index`

end index of flow

Definition at line 853 of file `vtss_phy_ts_api.h`.

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

- `vtss_api/include/vtss_phy_ts_api.h`

7.180 vtss_phy_ts_engine_action_t Struct Reference

Engine Action configuration options.

```
#include <vtss_phy_ts_api.h>
```

Data Fields

- `BOOL action_ptp`
- `BOOL action_gen`
- union {
 `vtss_phy_ts_ptp_engine_action_t ptp_conf [2]`
`vtss_phy_ts_oam_engine_action_t oam_conf [6]`
`vtss_phy_ts_generic_action_t gen_conf [6]`
} `action`

7.180.1 Detailed Description

Engine Action configuration options.

Note

Number of PTP actions in a engine depends on clock mode and delay method, like TC1Step and P2P, it requires 3 ingress rules to be added into PTP flows. There are total 6 flows in PTP and OAM comparator. For each frame type that needs to be timestamped requires one rule (i.e 1 flow). So application should decide the number of actions accordingly. For OAM only 2DM needs 2 flows, others needs 1 flow. The number of PTP/OAM actions config here is the maximum number, application should decide how many are valid for a engine based on clock mode and delay method.

Definition at line 1421 of file `vtss_phy_ts_api.h`.

7.180.2 Field Documentation

7.180.2.1 action_ptp

`BOOL vtss_phy_ts_engine_action_t::action_ptp`

is the action for PTP or OAM

Definition at line 1422 of file vtss_phy_ts_api.h.

7.180.2.2 action_gen

`BOOL vtss_phy_ts_engine_action_t::action_gen`

generic action or not

Definition at line 1423 of file vtss_phy_ts_api.h.

7.180.2.3 ptp_conf

`vtss_phy_ts_ptp_engine_action_t vtss_phy_ts_engine_action_t::ptp_conf[2]`

Max 2 PTP action per engine

Definition at line 1425 of file vtss_phy_ts_api.h.

7.180.2.4 oam_conf

`vtss_phy_ts_oam_engine_action_t vtss_phy_ts_engine_action_t::oam_conf[6]`

Max 6 OAM action per engine

Definition at line 1426 of file vtss_phy_ts_api.h.

7.180.2.5 gen_conf

`vtss_phy_ts_generic_action_t vtss_phy_ts_engine_action_t::gen_conf[6]`

Max 6 Generic action per engine

Definition at line 1427 of file vtss_phy_ts_api.h.

7.180.2.6 action

```
union { ... } vtss_phy_ts_engine_action_t::action
```

PTP/OAM action config

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

- [vtss_api/include/vtss_phy_ts_api.h](#)

7.181 vtss_phy_ts_engine_flow_conf_t Struct Reference

Analyzer flow configuration options.

```
#include <vtss_phy_ts_api.h>
```

Data Fields

- `BOOL eng_mode`
- `vtss_phy_ts_engine_channel_map_t channel_map [8]`
- `union {`
 - `vtss_phy_ts_ptp_engine_flow_conf_t ptp`
 - `vtss_phy_ts_oam_engine_flow_conf_t oam`
 - `vtss_phy_ts_generic_flow_conf_t gen``} flow_conf`

7.181.1 Detailed Description

Analyzer flow configuration options.

Note

Engine configuration will be parsed to know PTP or OAM flow based on encapsulation type provided during engine allocation.

Definition at line 1178 of file `vtss_phy_ts_api.h`.

7.181.2 Field Documentation

7.181.2.1 eng_mode

```
BOOL vtss_phy_ts_engine_flow_conf_t::eng_mode
```

engine enable/disable

Definition at line 1179 of file `vtss_phy_ts_api.h`.

7.181.2.2 channel_map

`vtss_phy_ts_engine_channel_map_t vtss_phy_ts_engine_flow_conf_t::channel_map[8]`

maps flows to channel for multi-channel timestamp block. flow_map can be set per comparator in HW

Definition at line 1180 of file vtss_phy_ts_api.h.

7.181.2.3 ptp

`vtss_phy_ts_ptp_engine_flow_conf_t vtss_phy_ts_engine_flow_conf_t::ptp`

PTP engine configuration

Definition at line 1183 of file vtss_phy_ts_api.h.

7.181.2.4 oam

`vtss_phy_ts_oam_engine_flow_conf_t vtss_phy_ts_engine_flow_conf_t::oam`

OAM engine configuration

Definition at line 1184 of file vtss_phy_ts_api.h.

7.181.2.5 gen

`vtss_phy_ts_generic_flow_conf_t vtss_phy_ts_engine_flow_conf_t::gen`

Generic match configuration

Definition at line 1185 of file vtss_phy_ts_api.h.

7.181.2.6 flow_conf

`union { ... } vtss_phy_ts_engine_flow_conf_t::flow_conf`

PTP/OAM flow config

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

- vtss_api/include/[vtss_phy_ts_api.h](#)

7.182 vtss_phy_ts_eth_conf_t Struct Reference

Analyzer Ethernet comparator configuration options.

```
#include <vtss_phy_ts_api.h>
```

Data Fields

- struct {
 BOOL pbb_en
 u16 etype
 u16 tpid
} comm_opt
- struct {
 BOOL flow_en
 u8 addr_match_mode
 u8 addr_match_select
 u8 mac_addr [6]
 BOOL vlan_check
 u8 num_tag
 u8 outer_tag_type
 u8 inner_tag_type
 u8 tag_range_mode
 union {
 struct {
 u16 upper
 u16 lower
 } range
 struct {
 u16 val
 u16 mask
 } value
 } outer_tag
 union {
 struct {
 u16 upper
 u16 lower
 } range
 struct {
 u16 val
 u16 mask
 } value
 struct {
 u32 val
 u32 mask
 } i_tag
 } inner_tag
} flow_opt [8]

7.182.1 Detailed Description

Analyzer Ethernet comparator configuration options.

Note

Common options apply all the flows within the comparator. Also there are per-flow configuration.

Definition at line 955 of file vtss_phy_ts_api.h.

7.182.2 Field Documentation

7.182.2.1 pbb_en

`BOOL vtss_phy_ts_eth_conf_t::pbb_en`

PBB tag present, not applicable for Eth2 comparator

Definition at line 957 of file vtss_phy_ts_api.h.

7.182.2.2 etype

`u16 vtss_phy_ts_eth_conf_t::etype`

The value of Ether type to be checked if Ethertype/length field is an Ethertype

Definition at line 958 of file vtss_phy_ts_api.h.

7.182.2.3 tpid

`u16 vtss_phy_ts_eth_conf_t::tpid`

VLAN TPID for S or B-tag

Definition at line 959 of file vtss_phy_ts_api.h.

7.182.2.4 comm_opt

`struct { ... } vtss_phy_ts_eth_conf_t::comm_opt`

Ethernet common config

7.182.2.5 flow_en

`BOOL vtss_phy_ts_eth_conf_t::flow_en`

flow enable/disable

Definition at line 963 of file vtss_phy_ts_api.h.

7.182.2.6 addr_match_mode

`u8 vtss_phy_ts_eth_conf_t::addr_match_mode`

Multiple match can be possible using OR

Definition at line 968 of file vtss_phy_ts_api.h.

7.182.2.7 addr_match_select

`u8 vtss_phy_ts_eth_conf_t::addr_match_select`

src or dest addr to be matched

Definition at line 972 of file vtss_phy_ts_api.h.

7.182.2.8 mac_addr

`u8 vtss_phy_ts_eth_conf_t::mac_addr[6]`

addr to be matched, src or dest

Definition at line 973 of file vtss_phy_ts_api.h.

7.182.2.9 vlan_check

`BOOL vtss_phy_ts_eth_conf_t::vlan_check`

TRUE=>verify configured VLAN tag configuration, FALSE=>parse VLAN tag if any, but don't check, for PBB I-tag is always checked

Definition at line 975 of file vtss_phy_ts_api.h.

7.182.2.10 num_tag

`u8 vtss_phy_ts_eth_conf_t::num_tag`

No of Tags (max 2 tag), for PBB at least I-tag should be present

Definition at line 976 of file vtss_phy_ts_api.h.

7.182.2.11 outer_tag_type

`u8 vtss_phy_ts_eth_conf_t::outer_tag_type`

for PBB enabled with 2-tag, this must be B-tag

Definition at line 981 of file vtss_phy_ts_api.h.

7.182.2.12 inner_tag_type

`u8 vtss_phy_ts_eth_conf_t::inner_tag_type`

for PBB this must be I-tag; also for single tag inner_tag is used

Definition at line 982 of file vtss_phy_ts_api.h.

7.182.2.13 tag_range_mode

`u8 vtss_phy_ts_eth_conf_t::tag_range_mode`

for PBB no range check is allowed

Definition at line 986 of file vtss_phy_ts_api.h.

7.182.2.14 upper

`u16 vtss_phy_ts_eth_conf_t::upper`

Upper value for outer tag range

Upper value for inner tag range

Definition at line 989 of file vtss_phy_ts_api.h.

7.182.2.15 lower

`u16 vtss_phy_ts_eth_conf_t::lower`

Lower value for outer tag range

Loower value for inner tag range

Definition at line 990 of file `vtss_phy_ts_api.h`.

7.182.2.16 range [1/2]

`struct { ... } vtss_phy_ts_eth_conf_t::range`

tag in range

7.182.2.17 val [1/2]

`u16 vtss_phy_ts_eth_conf_t::val`

Value

Definition at line 993 of file `vtss_phy_ts_api.h`.

7.182.2.18 mask [1/2]

`u16 vtss_phy_ts_eth_conf_t::mask`

Mask

Definition at line 994 of file `vtss_phy_ts_api.h`.

7.182.2.19 value [1/2]

`struct { ... } vtss_phy_ts_eth_conf_t::value`

tag in value/mask

7.182.2.20 outer_tag

`union { ... } vtss_phy_ts_eth_conf_t::outer_tag`

Outer tag

7.182.2.21 range [2/2]

```
struct { ... } vtss_phy_ts_eth_conf_t::range
```

tag in range

7.182.2.22 value [2/2]

```
struct { ... } vtss_phy_ts_eth_conf_t::value
```

tag in value/mask

7.182.2.23 val [2/2]

```
u32 vtss_phy_ts_eth_conf_t::val
```

24-bit I-tag value

Definition at line 1007 of file vtss_phy_ts_api.h.

7.182.2.24 mask [2/2]

```
u32 vtss_phy_ts_eth_conf_t::mask
```

24-bit I-tag mask

Definition at line 1008 of file vtss_phy_ts_api.h.

7.182.2.25 i_tag

```
struct { ... } vtss_phy_ts_eth_conf_t::i_tag
```

I-tag in value/mask. This is applicable for PBB i.e. Eth1 comparator

7.182.2.26 inner_tag

```
union { ... } vtss_phy_ts_eth_conf_t::inner_tag
```

Inner Tag

7.182.2.27 flow_opt

```
struct { ... } vtss_phy_ts_eth_conf_t::flow_opt[8]
```

Ethernet per flow config

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

- [vtss_api/include/vtss_phy_ts_api.h](#)

7.183 vtss_phy_ts_fifo_conf_t Struct Reference

Defines the params for FIFO SYNC function.

```
#include <vtss_phy_ts_api.h>
```

Data Fields

- `BOOL detect_only`
- `vtss_phy_ts_engine_t eng_recov`
- `vtss_phy_ts_engine_t eng_minE`
- `BOOL skip_rev_check`

7.183.1 Detailed Description

Defines the params for FIFO SYNC function.

Definition at line 2141 of file `vtss_phy_ts_api.h`.

7.183.2 Field Documentation

7.183.2.1 detect_only

```
BOOL vtss_phy_ts_fifo_conf_t::detect_only
```

TS FIFO OOS Detect only, no recovery, Only for Tesla

Definition at line 2142 of file `vtss_phy_ts_api.h`.

7.183.2.2 eng_recov

`vtss_phy_ts_engine_t vtss_phy_ts_fifo_conf_t::eng_recov`

Main Engine used for recovery, Only for Tesla

Definition at line 2143 of file vtss_phy_ts_api.h.

7.183.2.3 eng_minE

`vtss_phy_ts_engine_t vtss_phy_ts_fifo_conf_t::eng_minE`

Mini-E Engine used for recovery, Only for Tesla

Definition at line 2144 of file vtss_phy_ts_api.h.

7.183.2.4 skip_rev_check

`BOOL vtss_phy_ts_fifo_conf_t::skip_rev_check`

To force execution, regardless of revision

Definition at line 2145 of file vtss_phy_ts_api.h.

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

- vtss_api/include/vtss_phy_ts_api.h

7.184 vtss_phy_ts_fifo_sig_t Struct Reference

Tx TSFIFO entry signature.

```
#include <vtss_phy_ts_api.h>
```

Data Fields

- `vtss_phy_ts_fifo_sig_mask_t sig_mask`
- `u8 msg_type`
- `u8 domain_num`
- `u8 src_port_identity [10]`
- `u16 sequence_id`
- `u32 dest_ip`
- `u32 src_ip`
- `u8 dest_mac [6]`

7.184.1 Detailed Description

Tx TSFIFO entry signature.

Definition at line 669 of file vtss_phy_ts_api.h.

7.184.2 Field Documentation

7.184.2.1 sig_mask

`vtss_phy_ts_fifo_sig_mask_t` vtss_phy_ts_fifo_sig_t::sig_mask

valid signature fields

Definition at line 670 of file vtss_phy_ts_api.h.

7.184.2.2 msg_type

`u8` vtss_phy_ts_fifo_sig_t::msg_type

PTP message type

Definition at line 671 of file vtss_phy_ts_api.h.

7.184.2.3 domain_num

`u8` vtss_phy_ts_fifo_sig_t::domain_num

domain number in PTP message

Definition at line 672 of file vtss_phy_ts_api.h.

7.184.2.4 src_port_identity

`u8` vtss_phy_ts_fifo_sig_t::src_port_identity[10]

source port identity in PTP message

Definition at line 673 of file vtss_phy_ts_api.h.

7.184.2.5 sequence_id

```
u16 vtss_phy_ts_fifo_sig_t::sequence_id
```

PTP message sequence ID

Definition at line 674 of file vtss_phy_ts_api.h.

7.184.2.6 dest_ip

```
u32 vtss_phy_ts_fifo_sig_t::dest_ip
```

Destination IP

Definition at line 675 of file vtss_phy_ts_api.h.

7.184.2.7 src_ip

```
u32 vtss_phy_ts_fifo_sig_t::src_ip
```

Source IP

Definition at line 676 of file vtss_phy_ts_api.h.

7.184.2.8 dest_mac

```
u8 vtss_phy_ts_fifo_sig_t::dest_mac[6]
```

Destination MAC

Definition at line 677 of file vtss_phy_ts_api.h.

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

- vtss_api/include/[vtss_phy_ts_api.h](#)

7.185 vtss_phy_ts_gen_conf_t Struct Reference

Analyzer Generic data configuration options using IP comparator.

```
#include <vtss_phy_ts_api.h>
```

Data Fields

- struct {
 u8 flow_offset
 u8 next_prot_offset
} comm_opt

- struct {
 BOOL flow_en
 u32 data [4]
 u32 mask [4]
} flow_opt [8]

7.185.1 Detailed Description

Analyzer Generic data configuration options using IP comparator.

Definition at line 1122 of file vtss_phy_ts_api.h.

7.185.2 Field Documentation

7.185.2.1 flow_offset

u8 vtss_phy_ts_gen_conf_t::flow_offset

Offset of data pattern to match with current comparator

Definition at line 1124 of file vtss_phy_ts_api.h.

7.185.2.2 next_prot_offset

u8 vtss_phy_ts_gen_conf_t::next_prot_offset

Offset of data pattern to match with next comparator

Definition at line 1125 of file vtss_phy_ts_api.h.

7.185.2.3 comm_opt

struct { ... } vtss_phy_ts_gen_conf_t::comm_opt

Generic Matching common configuration

7.185.2.4 flow_en

`BOOL vtss_phy_ts_gen_conf_t::flow_en`

Enable the flow

Definition at line 1129 of file vtss_phy_ts_api.h.

7.185.2.5 data

`u32 vtss_phy_ts_gen_conf_t::data[4]`

Data byte pattern to match

Definition at line 1130 of file vtss_phy_ts_api.h.

7.185.2.6 mask

`u32 vtss_phy_ts_gen_conf_t::mask[4]`

Mask of the matching pattern

Definition at line 1131 of file vtss_phy_ts_api.h.

7.185.2.7 flow_opt

`struct { ... } vtss_phy_ts_gen_conf_t::flow_opt[8]`

Generic matching config per flow

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

- vtss_api/include/vtss_phy_ts_api.h

7.186 vtss_phy_ts_generic_action_t Struct Reference

Generic Action configuration option.

```
#include <vtss_phy_ts_api.h>
```

Data Fields

- `BOOL enable`
- `vtss_phy_ts_engine_channel_map_t channel_map`
- `u8 flow_id`
- `u32 data [2]`
- `u32 mask [2]`
- `vtss_phy_ts_action_format ts_type`
- `u32 ts_offset`

7.186.1 Detailed Description

Generic Action configuration option.

Definition at line 1400 of file `vtss_phy_ts_api.h`.

7.186.2 Field Documentation

7.186.2.1 enable

`BOOL vtss_phy_ts_generic_action_t::enable`

Generic action active/enable or not

Definition at line 1401 of file `vtss_phy_ts_api.h`.

7.186.2.2 channel_map

`vtss_phy_ts_engine_channel_map_t vtss_phy_ts_generic_action_t::channel_map`

maps action to channel for multi-channel timestamp block

Definition at line 1402 of file `vtss_phy_ts_api.h`.

7.186.2.3 flow_id

`u8 vtss_phy_ts_generic_action_t::flow_id`

Flow id to be associated with this action

Definition at line 1403 of file `vtss_phy_ts_api.h`.

7.186.2.4 data

`u32 vtss_phy_ts_generic_action_t::data[2]`

Matching data pattern

Definition at line 1404 of file vtss_phy_ts_api.h.

7.186.2.5 mask

`u32 vtss_phy_ts_generic_action_t::mask[2]`

Mask for the matching pattern

Definition at line 1405 of file vtss_phy_ts_api.h.

7.186.2.6 ts_type

`vtss_phy_ts_action_format vtss_phy_ts_generic_action_t::ts_type`

Timestamp type 4-byte or 10 byte timestamp

Definition at line 1406 of file vtss_phy_ts_api.h.

7.186.2.7 ts_offset

`u32 vtss_phy_ts_generic_action_t::ts_offset`

Timestamp offset

Definition at line 1407 of file vtss_phy_ts_api.h.

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

- vtss_api/include/[vtss_phy_ts_api.h](#)

7.187 vtss_phy_ts_generic_flow_conf_t Struct Reference

Generic engine flow configuration options.

```
#include <vtss_phy_ts_api.h>
```

Data Fields

- [vtss_phy_ts_eth_conf_t eth1_opt](#)
- [vtss_phy_ts_gen_conf_t gen_opt](#)

7.187.1 Detailed Description

Generic engine flow configuration options.

Definition at line 1168 of file vtss_phy_ts_api.h.

7.187.2 Field Documentation

7.187.2.1 eth1_opt

[vtss_phy_ts_eth_conf_t](#) vtss_phy_ts_generic_flow_conf_t::eth1_opt

Eth-1 comparator

Definition at line 1169 of file vtss_phy_ts_api.h.

7.187.2.2 gen_opt

[vtss_phy_ts_gen_conf_t](#) vtss_phy_ts_generic_flow_conf_t::gen_opt

Generic : It uses IP1 comparator

Definition at line 1170 of file vtss_phy_ts_api.h.

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

- vtss_api/include/[vtss_phy_ts_api.h](#)

7.188 vtss_phy_ts_ietf_mpls_ach_oam_conf_t Struct Reference

Analyzer OAM comparator, IETF MPLS ACH OAM Packet format configuration options.

```
#include <vtss_phy_ts_api.h>
```

Data Fields

- [vtss_phy_ts_ietf_mpls_ach_oam_delaym_type_t delaym_type](#)
- [vtss_phy_ts_ietf_mpls_ach_oam_ts_format_t ts_format](#)
- [u8 ds](#)

7.188.1 Detailed Description

Analyzer OAM comparator, IETF MPLS ACH OAM Packet format configuration options.

Definition at line 1368 of file `vtss_phy_ts_api.h`.

7.188.2 Field Documentation

7.188.2.1 `delaym_type`

`vtss_phy_ts_ietf_mpls_ach_oam_delaym_type_t` `vtss_phy_ts_ietf_mpls_ach_oam_conf_t::delaym_type`

OAM delay measurement method

Definition at line 1369 of file `vtss_phy_ts_api.h`.

7.188.2.2 `ts_format`

`vtss_phy_ts_ietf_mpls_ach_oam_ts_format_t` `vtss_phy_ts_ietf_mpls_ach_oam_conf_t::ts_format`

OAM DM Timestamp format

Definition at line 1370 of file `vtss_phy_ts_api.h`.

7.188.2.3 `ds`

`u8` `vtss_phy_ts_ietf_mpls_ach_oam_conf_t::ds`

DSCP value, that corresponds to a traffic class being measured.

Definition at line 1371 of file `vtss_phy_ts_api.h`.

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

- `vtss_api/include/vtss_phy_ts_api.h`

7.189 `vtss_phy_ts_init_conf_t` Struct Reference

Defines the initial parameters to be passed to init function.

```
#include <vtss_phy_ts_api.h>
```

Data Fields

- `vtss_phy_ts_clockfreq_t clk_freq`
- `vtss_phy_ts_clock_src_t clk_src`
- `vtss_phy_ts_rxtimestamp_pos_t rx_ts_pos`
- `vtss_phy_ts_rxtimestamp_len_t rx_ts_len`
- `vtss_phy_ts_fifo_mode_t tx_fifo_mode`
- `vtss_phy_ts_fifo_timestamp_len_t tx_ts_len`
- `BOOL tx_fifo_spi_conf`
- `u8 tx_fifo_hi_clk_cycs`
- `u8 tx_fifo_lo_clk_cycs`
- `vtss_phy_ts_8487_xaui_sel_t xaui_sel_8487`
- `vtss_phy_ts_tc_op_mode_t tc_op_mode`
- `BOOL auto_clear_ls`
- `BOOL macsec_ena`
- `BOOL chk_ing_modified`
- `BOOL one_step_txfifo`

7.189.1 Detailed Description

Defines the initial parameters to be passed to init function.

Definition at line 1743 of file vtss_phy_ts_api.h.

7.189.2 Field Documentation

7.189.2.1 clk_freq

`vtss_phy_ts_clockfreq_t vtss_phy_ts_init_conf_t::clk_freq`

reference clock frequency

Definition at line 1744 of file vtss_phy_ts_api.h.

7.189.2.2 clk_src

`vtss_phy_ts_clock_src_t vtss_phy_ts_init_conf_t::clk_src`

clock source

Definition at line 1745 of file vtss_phy_ts_api.h.

7.189.2.3 rx_ts_pos

```
vtss_phy_ts_rxtimestamp_pos_t vtss_phy_ts_init_conf_t::rx_ts_pos
```

Rx timestamp position

Definition at line 1746 of file vtss_phy_ts_api.h.

7.189.2.4 rx_ts_len

```
vtss_phy_ts_rxtimestamp_len_t vtss_phy_ts_init_conf_t::rx_ts_len
```

Rx timestamp length

Definition at line 1747 of file vtss_phy_ts_api.h.

7.189.2.5 tx_fifo_mode

```
vtss_phy_ts_fifo_mode_t vtss_phy_ts_init_conf_t::tx_fifo_mode
```

Tx TSFIFO access mode

Definition at line 1748 of file vtss_phy_ts_api.h.

7.189.2.6 tx_ts_len

```
vtss_phy_ts_fifo_timestamp_len_t vtss_phy_ts_init_conf_t::tx_ts_len
```

timestamp size in Tx TSFIFO

Definition at line 1749 of file vtss_phy_ts_api.h.

7.189.2.7 tx_fifo_spi_conf

```
BOOL vtss_phy_ts_init_conf_t::tx_fifo_spi_conf
```

Modify default 1588_spi configuration, applicable only on PHYs with SPI timestamp fifo support

Definition at line 1750 of file vtss_phy_ts_api.h.

7.189.2.8 tx_fifo_hi_clk_cycs

`u8 vtss_phy_ts_init_conf_t::tx_fifo_hi_clk_cycs`

Number of clock periods that the spi_clk is high

Definition at line 1751 of file vtss_phy_ts_api.h.

7.189.2.9 tx_fifo_lo_clk_cycs

`u8 vtss_phy_ts_init_conf_t::tx_fifo_lo_clk_cycs`

Number of clock periods that the spi_clk is low

Definition at line 1752 of file vtss_phy_ts_api.h.

7.189.2.10 xaui_sel_8487

`vtss_phy_ts_8487_xaui_sel_t vtss_phy_ts_init_conf_t::xaui_sel_8487`

8487 XAUI lane selection

Definition at line 1754 of file vtss_phy_ts_api.h.

7.189.2.11 tc_op_mode

`vtss_phy_ts_tc_op_mode_t vtss_phy_ts_init_conf_t::tc_op_mode`

TC operating mode

Definition at line 1759 of file vtss_phy_ts_api.h.

7.189.2.12 auto_clear_ls

`BOOL vtss_phy_ts_init_conf_t::auto_clear_ls`

Load and Save of LTC are auto cleared

Definition at line 1760 of file vtss_phy_ts_api.h.

7.189.2.13 macsec_ena

`BOOL vtss_phy_ts_init_conf_t::macsec_ena`

MACsec is enabled or disabled

Definition at line 1761 of file `vtss_phy_ts_api.h`.

7.189.2.14 chk_ing_modified

`BOOL vtss_phy_ts_init_conf_t::chk_ing_modified`

True if the flag bit needs to be modified in ingress and thus in egress

Definition at line 1762 of file `vtss_phy_ts_api.h`.

7.189.2.15 one_step_txfifo

`BOOL vtss_phy_ts_init_conf_t::one_step_txfifo`

used when transmitting Delay_Req in one step mode. FALSE when correctionfield update is used instead

Definition at line 1763 of file `vtss_phy_ts_api.h`.

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

- `vtss_api/include/vtss_phy_ts_api.h`

7.190 **vtss_phy_ts_ip_conf_t** Struct Reference

Analyzer IP comparator configuration options.

```
#include <vtss_phy_ts_api.h>
```

Data Fields

```

• struct {
    u8 ip_mode
    u16 sport_val
    u16 sport_mask
    u16 dport_val
    u16 dport_mask
} comm_opt

• struct {
    BOOL flow_en
    u8 match_mode
    union {
        struct {
            u32 addr
            u32 mask
        } ipv4
        struct {
            u32 addr [4]
            u32 mask [4]
        } ipv6
    } ip_addr
} flow_opt [8]

```

7.190.1 Detailed Description

Analyzer IP comparator configuration options.

Note

Common options apply all the flows within the comparator. Also there are per flow configuration.

Definition at line 1019 of file vtss_phy_ts_api.h.

7.190.2 Field Documentation

7.190.2.1 ip_mode

```
u8 vtss_phy_ts_ip_conf_t::ip_mode
```

IPv4, IPv6 if next protocol is not UDP, next UDP fields are not used

Definition at line 1023 of file vtss_phy_ts_api.h.

7.190.2.2 sport_val

`u16 vtss_phy_ts_ip_conf_t::sport_val`

UDP source port value

Definition at line 1025 of file vtss_phy_ts_api.h.

7.190.2.3 sport_mask

`u16 vtss_phy_ts_ip_conf_t::sport_mask`

UDP source port mask

Definition at line 1026 of file vtss_phy_ts_api.h.

7.190.2.4 dport_val

`u16 vtss_phy_ts_ip_conf_t::dport_val`

UDP dest port value

Definition at line 1027 of file vtss_phy_ts_api.h.

7.190.2.5 dport_mask

`u16 vtss_phy_ts_ip_conf_t::dport_mask`

UDP dest port mask

Definition at line 1028 of file vtss_phy_ts_api.h.

7.190.2.6 comm_opt

`struct { ... } vtss_phy_ts_ip_conf_t::comm_opt`

IP common config

7.190.2.7 flow_en

`BOOL vtss_phy_ts_ip_conf_t::flow_en`

flow enable/disable

Definition at line 1032 of file `vtss_phy_ts_api.h`.

7.190.2.8 match_mode

`u8 vtss_phy_ts_ip_conf_t::match_mode`

match src, dest or either IP address

Definition at line 1036 of file `vtss_phy_ts_api.h`.

7.190.2.9 addr

`u32 vtss_phy_ts_ip_conf_t::addr[4]`

IPv4 address

IPv6 Address

Definition at line 1039 of file `vtss_phy_ts_api.h`.

7.190.2.10 mask

`u32 vtss_phy_ts_ip_conf_t::mask[4]`

IPv4 address mask

IPv6 Mask

Definition at line 1040 of file `vtss_phy_ts_api.h`.

7.190.2.11 ipv4

`struct { ... } vtss_phy_ts_ip_conf_t::ipv4`

IPv4 Address

7.190.2.12 ipv6

```
struct { ... } vtss_phy_ts_ip_conf_t::ipv6
```

IPv6 Mask

7.190.2.13 ip_addr

```
union { ... } vtss_phy_ts_ip_conf_t::ip_addr
```

IPv4/IPv6 address to be matched

7.190.2.14 flow_opt

```
struct { ... } vtss_phy_ts_ip_conf_t::flow_opt[8]
```

IP per flow config

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

- [vtss_api/include/vtss_phy_ts_api.h](#)

7.191 vtss_phy_ts_mpls_conf_t Struct Reference

Analyzer MPLS comparator configuration options.

```
#include <vtss_phy_ts_api.h>
```

Data Fields

- struct {

 BOOL cw_en
 } comm_opt
- struct {

 BOOL flow_en
 u8 stack_depth
 u8 stack_ref_point
 union {
 struct {
 vtss_phy_ts_mpls_lvl_rng_t top
 vtss_phy_ts_mpls_lvl_rng_t frst_lvl_after_top
 vtss_phy_ts_mpls_lvl_rng_t snd_lvl_after_top
 vtss_phy_ts_mpls_lvl_rng_t thrd_lvl_after_top
 } top_down
 struct {
 vtss_phy_ts_mpls_lvl_rng_t end
 vtss_phy_ts_mpls_lvl_rng_t frst_lvl_before_end
 vtss_phy_ts_mpls_lvl_rng_t snd_lvl_before_end
 vtss_phy_ts_mpls_lvl_rng_t thrd_lvl_before_end
 } bottom_up
 } stack_level
 } flow_opt [8]

7.191.1 Detailed Description

Analyzer MPLS comparator configuration options.

Definition at line 1061 of file vtss_phy_ts_api.h.

7.191.2 Field Documentation

7.191.2.1 cw_en

`BOOL` `vtss_phy_ts_mpls_conf_t::cw_en`

flow uses psudowire control word or not

Definition at line 1063 of file vtss_phy_ts_api.h.

7.191.2.2 comm_opt

`struct { ... } vtss_phy_ts_mpls_conf_t::comm_opt`

MPLS common config

7.191.2.3 flow_en

`BOOL` `vtss_phy_ts_mpls_conf_t::flow_en`

flow enable/disable

Definition at line 1066 of file vtss_phy_ts_api.h.

7.191.2.4 stack_depth

`u8` `vtss_phy_ts_mpls_conf_t::stack_depth`

depth of MPLS level; multiple depth match can be possible using OR

Definition at line 1072 of file vtss_phy_ts_api.h.

7.191.2.5 stack_ref_point

`u8 vtss_phy_ts_mpls_conf_t::stack_ref_point`

Search direction for label matching: top to bottom or bottom to top

Definition at line 1076 of file `vtss_phy_ts_api.h`.

7.191.2.6 top

`vtss_phy_ts_mpls_lvl_rng_t vtss_phy_ts_mpls_conf_t::top`

Top level

Definition at line 1079 of file `vtss_phy_ts_api.h`.

7.191.2.7 frst_lvl_after_top

`vtss_phy_ts_mpls_lvl_rng_t vtss_phy_ts_mpls_conf_t::frst_lvl_after_top`

First label after the top label

Definition at line 1080 of file `vtss_phy_ts_api.h`.

7.191.2.8 snd_lvl_after_top

`vtss_phy_ts_mpls_lvl_rng_t vtss_phy_ts_mpls_conf_t::snd_lvl_after_top`

Second label after the top label

Definition at line 1081 of file `vtss_phy_ts_api.h`.

7.191.2.9 thrd_lvl_after_top

`vtss_phy_ts_mpls_lvl_rng_t vtss_phy_ts_mpls_conf_t::thrd_lvl_after_top`

Third label after the top label

Definition at line 1082 of file `vtss_phy_ts_api.h`.

7.191.2.10 top_down

```
struct { ... } vtss_phy_ts_mpls_conf_t::top_down
```

Top down configuration

7.191.2.11 end

```
vtss_phy_ts_mpls_lvl_rng_t vtss_phy_ts_mpls_conf_t::end
```

End level

Definition at line 1085 of file vtss_phy_ts_api.h.

7.191.2.12 frst_lvl_before_end

```
vtss_phy_ts_mpls_lvl_rng_t vtss_phy_ts_mpls_conf_t::frst_lvl_before_end
```

First label before the end label

Definition at line 1086 of file vtss_phy_ts_api.h.

7.191.2.13 snd_lvl_before_end

```
vtss_phy_ts_mpls_lvl_rng_t vtss_phy_ts_mpls_conf_t::snd_lvl_before_end
```

Second label before the end label

Definition at line 1087 of file vtss_phy_ts_api.h.

7.191.2.14 thrd_lvl_before_end

```
vtss_phy_ts_mpls_lvl_rng_t vtss_phy_ts_mpls_conf_t::thrd_lvl_before_end
```

Third label before the end label

Definition at line 1088 of file vtss_phy_ts_api.h.

7.191.2.15 bottom_up

```
struct { ... } vtss_phy_ts_mpls_conf_t::bottom_up
```

Bottom up configuration

7.191.2.16 stack_level

```
union { ... } vtss_phy_ts_mpls_conf_t::stack_level
```

4 level values; top_down or bottom_up depends on stack_ref_point

7.191.2.17 flow_opt

```
struct { ... } vtss_phy_ts_mpls_conf_t::flow_opt[8]
```

MPLS per flow config

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

- [vtss_api/include/vtss_phy_ts_api.h](#)

7.192 vtss_phy_ts_mpls_lvl_rng_t Struct Reference

MPLS level range.

```
#include <vtss_phy_ts_api.h>
```

Data Fields

- [u32 lower](#)
- [u32 upper](#)

7.192.1 Detailed Description

MPLS level range.

Definition at line 1053 of file [vtss_phy_ts_api.h](#).

7.192.2 Field Documentation

7.192.2.1 lower

```
u32 vtss_phy_ts_mpls_lvl_rng_t::lower
```

lower range value

Definition at line 1054 of file [vtss_phy_ts_api.h](#).

7.192.2.2 upper

`u32 vtss_phy_ts_mpls_lvl_rng_t::upper`

upper range value

Definition at line 1055 of file `vtss_phy_ts_api.h`.

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

- `vtss_api/include/vtss_phy_ts_api.h`

7.193 `vtss_phy_ts_nphase_status_t` Struct Reference

n-phase status

```
#include <vtss_phy_ts_api.h>
```

Data Fields

- `BOOL enable`
- `BOOL CALIB_ERR`
- `BOOL CALIB_DONE`

7.193.1 Detailed Description

n-phase status

Definition at line 1885 of file `vtss_phy_ts_api.h`.

7.193.2 Field Documentation

7.193.2.1 enable

`BOOL vtss_phy_ts_nphase_status_t::enable`

Enabled status

Definition at line 1886 of file `vtss_phy_ts_api.h`.

7.193.2.2 CALIB_ERR

`BOOL vtss_phy_ts_nphase_status_t::CALIB_ERR`

Calibration error

Definition at line 1887 of file vtss_phy_ts_api.h.

7.193.2.3 CALIB_DONE

`BOOL vtss_phy_ts_nphase_status_t::CALIB_DONE`

Calibration done

Definition at line 1888 of file vtss_phy_ts_api.h.

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

- `vtss_api/include/vtss_phy_ts_api.h`

7.194 vtss_phy_ts_oam_engine_action_t Struct Reference

OAM Action configuration options.

```
#include <vtss_phy_ts_api.h>
```

Data Fields

- `BOOL enable`
- `BOOL y1731_en`
- `vtss_phy_ts_engine_channel_map_t channel_map`
- `u8 version`
- `union {`
 - `vtss_phy_ts_y1731_oam_conf_t y1731_oam_conf`
 - `vtss_phy_ts_ietf_mpls_ach_oam_conf_t ietf_oam_conf``}` `oam_conf`

7.194.1 Detailed Description

OAM Action configuration options.

Note

Timestamp action will be based on OAM delay measurement method.

Definition at line 1378 of file vtss_phy_ts_api.h.

7.194.2 Field Documentation

7.194.2.1 enable

`BOOL vtss_phy_ts_oam_engine_action_t::enable`

OAM action active/enable or not

Definition at line 1379 of file vtss_phy_ts_api.h.

7.194.2.2 y1731_en

`BOOL vtss_phy_ts_oam_engine_action_t::y1731_en`

Y.1731 Message Format Enabled/Disable

Definition at line 1380 of file vtss_phy_ts_api.h.

7.194.2.3 channel_map

`vtss_phy_ts_engine_channel_map_t vtss_phy_ts_oam_engine_action_t::channel_map`

maps action to channel for multi-channel timestamp block

Definition at line 1381 of file vtss_phy_ts_api.h.

7.194.2.4 version

`u8 vtss_phy_ts_oam_engine_action_t::version`

Protocol Version; only 0 is supported

Definition at line 1382 of file vtss_phy_ts_api.h.

7.194.2.5 y1731_oam_conf

`vtss_phy_ts_y1731_oam_conf_t vtss_phy_ts_oam_engine_action_t::y1731_oam_conf`

Y.1731 OAM configuration

Definition at line 1384 of file vtss_phy_ts_api.h.

7.194.2.6 ietf_oam_conf

`vtss_phy_ts_ietf_mpls_ach_oam_conf_t` `vtss_phy_ts_oam_engine_action_t::ietf_oam_conf`

IETF OAM configuration

Definition at line 1385 of file `vtss_phy_ts_api.h`.

7.194.2.7 oam_conf

`union { ... } vtss_phy_ts_oam_engine_action_t::oam_conf`

OAM action config

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

- `vtss_api/include/vtss_phy_ts_api.h`

7.195 vtss_phy_ts_oam_engine_flow_conf_t Struct Reference

OAM engine flow configuration options.

```
#include <vtss_phy_ts_api.h>
```

Data Fields

- `vtss_phy_ts_eth_conf_t eth1_opt`
- `vtss_phy_ts_eth_conf_t eth2_opt`
- `vtss_phy_ts_mpls_conf_t mpls_opt`
- `vtss_phy_ts_ach_conf_t ach_opt`

7.195.1 Detailed Description

OAM engine flow configuration options.

Definition at line 1158 of file `vtss_phy_ts_api.h`.

7.195.2 Field Documentation

7.195.2.1 eth1_opt

`vtss_phy_ts_eth_conf_t vtss_phy_ts_oam_engine_flow_conf_t::eth1_opt`

Eth-1 comparator

Definition at line 1159 of file `vtss_phy_ts_api.h`.

7.195.2.2 eth2_opt

`vtss_phy_ts_eth_conf_t vtss_phy_ts_oam_engine_flow_conf_t::eth2_opt`

Eth-2 comparator; for single ETH encapsulation, Eth-1 is used

Definition at line 1160 of file `vtss_phy_ts_api.h`.

7.195.2.3 mpls_opt

`vtss_phy_ts_mpls_conf_t vtss_phy_ts_oam_engine_flow_conf_t::mpls_opt`

MPLS comparator

Definition at line 1161 of file `vtss_phy_ts_api.h`.

7.195.2.4 ach_opt

`vtss_phy_ts_ach_conf_t vtss_phy_ts_oam_engine_flow_conf_t::ach_opt`

ACH: it uses the IP1 comparator, so IP1 and ACH can not be enabled simultaneously

Definition at line 1162 of file `vtss_phy_ts_api.h`.

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

- `vtss_api/include/vtss_phy_ts_api.h`

7.196 vtss_phy_ts_pps_config_s Struct Reference

PPS Configuration.

```
#include <vtss_phy_ts_api.h>
```

Data Fields

- `u32 pps_width_adj`
- `u32 pps_offset`
- `u32 pps_output_enable`

7.196.1 Detailed Description

PPS Configuration.

Definition at line 102 of file `vtss_phy_ts_api.h`.

7.196.2 Field Documentation

7.196.2.1 pps_width_adj

`u32 vtss_phy_ts_pps_config_s::pps_width_adj`

The value of nano second counter upto which 1PPS is held high

Definition at line 103 of file `vtss_phy_ts_api.h`.

7.196.2.2 pps_offset

`u32 vtss_phy_ts_pps_config_s::pps_offset`

PPS pulse offset in nano seconds

Definition at line 104 of file `vtss_phy_ts_api.h`.

7.196.2.3 pps_output_enable

`u32 vtss_phy_ts_pps_config_s::pps_output_enable`

PPS pulse output is enabled for this port

Definition at line 105 of file `vtss_phy_ts_api.h`.

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

- `vtss_api/include/vtss_phy_ts_api.h`

7.197 vtss_phy_ts_ptp_conf_t Struct Reference

Analyzer PTP comparator configuration options.

```
#include <vtss_phy_ts_api.h>
```

Data Fields

- `BOOL range_en`
- `union {`
- `struct {`
- `u8 val`
- `u8 mask`
- `} value`
- `struct {`
- `u8 upper`
- `u8 lower`
- `} range`
- `} domain`

7.197.1 Detailed Description

Analyzer PTP comparator configuration options.

Definition at line 1268 of file vtss_phy_ts_api.h.

7.197.2 Field Documentation

7.197.2.1 range_en

```
BOOL vtss_phy_ts_ptp_conf_t::range_en
```

PTP domain number in range enable/disable

Definition at line 1269 of file vtss_phy_ts_api.h.

7.197.2.2 val

```
u8 vtss_phy_ts_ptp_conf_t::val
```

PTP domain number value

Definition at line 1272 of file vtss_phy_ts_api.h.

7.197.2.3 mask

```
u8 vtss_phy_ts_ptp_conf_t::mask
```

PTP domain number mask

Definition at line 1273 of file vtss_phy_ts_api.h.

7.197.2.4 value

```
struct { ... } vtss_phy_ts_ptp_conf_t::value
```

specific PTP domain, for don't care set mask as '0'

7.197.2.5 upper

```
u8 vtss_phy_ts_ptp_conf_t::upper
```

Ranger upper value

Definition at line 1276 of file vtss_phy_ts_api.h.

7.197.2.6 lower

```
u8 vtss_phy_ts_ptp_conf_t::lower
```

Range lower value

Definition at line 1277 of file vtss_phy_ts_api.h.

7.197.2.7 range

```
struct { ... } vtss_phy_ts_ptp_conf_t::range
```

PTP domain range configuration

7.197.2.8 domain

```
union { ... } vtss_phy_ts_ptp_conf_t::domain
```

PTP domain number configuration

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

- vtss_api/include/vtss_phy_ts_api.h

7.198 vtss_phy_ts_ptp_engine_action_t Struct Reference

Analyzer PTP action configuration options.

```
#include <vtss_phy_ts_api.h>
```

Data Fields

- `BOOL enable`
- `vtss_phy_ts_engine_channel_map_t channel_map`
- `vtss_phy_ts_ptp_conf_t ptp_conf`
- `vtss_phy_ts_ptp_clock_mode_t clk_mode`
- `vtss_phy_ts_ptp_delaym_type_t delaym_type`
- `BOOL cf_update`

7.198.1 Detailed Description

Analyzer PTP action configuration options.

Note

Timestamp action will be based on clock type and delay measurement method.

Definition at line 1310 of file vtss_phy_ts_api.h.

7.198.2 Field Documentation

7.198.2.1 enable

```
BOOL vtss_phy_ts_ptp_engine_action_t::enable
```

PTP action active/enable or not

Definition at line 1311 of file vtss_phy_ts_api.h.

7.198.2.2 channel_map

```
vtss_phy_ts_engine_channel_map_t vtss_phy_ts_ptp_engine_action_t::channel_map
```

maps action to channel for multi-channel timestamp block

Definition at line 1312 of file vtss_phy_ts_api.h.

7.198.2.3 ptp_conf

`vtss_phy_ts_ptp_conf_t` `vtss_phy_ts_ptp_engine_action_t::ptp_conf`

PTP configuration

Definition at line 1313 of file `vtss_phy_ts_api.h`.

7.198.2.4 clk_mode

`vtss_phy_ts_ptp_clock_mode_t` `vtss_phy_ts_ptp_engine_action_t::clk_mode`

clock mode: bc1step, bc2step, tc1step or tc2step

Definition at line 1314 of file `vtss_phy_ts_api.h`.

7.198.2.5 delaym_type

`vtss_phy_ts_ptp_delaym_type_t` `vtss_phy_ts_ptp_engine_action_t::delaym_type`

delay measurement method: P2P, E2E

Definition at line 1315 of file `vtss_phy_ts_api.h`.

7.198.2.6 cf_update

`BOOL` `vtss_phy_ts_ptp_engine_action_t::cf_update`

correction field update for bc1step

Definition at line 1316 of file `vtss_phy_ts_api.h`.

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

- `vtss_api/include/vtss_phy_ts_api.h`

7.199 **vtss_phy_ts_ptp_engine_flow_conf_t** Struct Reference

PTP engine flow configuration options.

```
#include <vtss_phy_ts_api.h>
```

Data Fields

- `vtss_phy_ts_eth_conf_t eth1_opt`
- `vtss_phy_ts_eth_conf_t eth2_opt`
- `vtss_phy_ts_ip_conf_t ip1_opt`
- `vtss_phy_ts_ip_conf_t ip2_opt`
- `vtss_phy_ts_mpls_conf_t mpls_opt`
- `vtss_phy_ts_ach_conf_t ach_opt`

7.199.1 Detailed Description

PTP engine flow configuration options.

Definition at line 1146 of file `vtss_phy_ts_api.h`.

7.199.2 Field Documentation

7.199.2.1 eth1_opt

`vtss_phy_ts_eth_conf_t vtss_phy_ts_ptp_engine_flow_conf_t::eth1_opt`

Eth-1 comparator

Definition at line 1147 of file `vtss_phy_ts_api.h`.

7.199.2.2 eth2_opt

`vtss_phy_ts_eth_conf_t vtss_phy_ts_ptp_engine_flow_conf_t::eth2_opt`

Eth-2 comparator; for single ETH encapsulation, Eth-1 is used

Definition at line 1148 of file `vtss_phy_ts_api.h`.

7.199.2.3 ip1_opt

`vtss_phy_ts_ip_conf_t vtss_phy_ts_ptp_engine_flow_conf_t::ip1_opt`

IP-1 comparator

Definition at line 1149 of file `vtss_phy_ts_api.h`.

7.199.2.4 ip2_opt

`vtss_phy_ts_ip_conf_t` `vtss_phy_ts_ptp_engine_flow_conf_t::ip2_opt`

IP-2 comparator; for single IP encapsulation, IP-1 is used

Definition at line 1150 of file `vtss_phy_ts_api.h`.

7.199.2.5 mpls_opt

`vtss_phy_ts_mpls_conf_t` `vtss_phy_ts_ptp_engine_flow_conf_t::mpls_opt`

MPLS comparator

Definition at line 1151 of file `vtss_phy_ts_api.h`.

7.199.2.6 ach_opt

`vtss_phy_ts_ach_conf_t` `vtss_phy_ts_ptp_engine_flow_conf_t::ach_opt`

ACH: it uses the IP1 comparator, so IP1 and ACH can not be enabled simultaneously

Definition at line 1152 of file `vtss_phy_ts_api.h`.

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

- `vtss_api/include/vtss_phy_ts_api.h`

7.200 vtss_phy_ts_sertod_conf_t Struct Reference

PHY timestamp in seconds and nanoseconds (10 bytes Timestamp)

```
#include <vtss_phy_ts_api.h>
```

Data Fields

- `BOOL ip_enable`
- `BOOL op_enable`
- `BOOL ls_inv`

7.200.1 Detailed Description

PHY timestamp in seconds and nanoseconds (10 bytes Timestamp)

Definition at line 330 of file `vtss_phy_ts_api.h`.

7.200.2 Field Documentation

7.200.2.1 ip_enable

`BOOL vtss_phy_ts_sertod_conf_t::ip_enable`

Serial ToD Input Enable

Definition at line 331 of file `vtss_phy_ts_api.h`.

7.200.2.2 op_enable

`BOOL vtss_phy_ts_sertod_conf_t::op_enable`

Serial ToD Output Enable

Definition at line 332 of file `vtss_phy_ts_api.h`.

7.200.2.3 ls_inv

`BOOL vtss_phy_ts_sertod_conf_t::ls_inv`

Invert the polarity of Load Save

Definition at line 333 of file `vtss_phy_ts_api.h`.

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

- `vtss_api/include/vtss_phy_ts_api.h`

7.201 `vtss_phy_ts_stats_t` Struct Reference

Timestamping Statistics.

```
#include <vtss_phy_ts_api.h>
```

Data Fields

- `u32 ingr_pream_shrink_err`
- `u32 egr_pream_shrink_err`
- `u32 ingr_fcs_err`
- `u32 egr_fcs_err`
- `u32 ingr_frm_mod_cnt`
- `u32 egr_frm_mod_cnt`
- `u32 ts_fifo_tx_cnt`
- `u32 ts_fifo_drop_cnt`

7.201.1 Detailed Description

Timestamping Statistics.

Note

Use [vtss_phy_ts_stats_get\(\)](#) to retrieve current statistics.

Definition at line 1574 of file vtss_phy_ts_api.h.

7.201.2 Field Documentation

7.201.2.1 ingr_pream_shrink_err

```
u32 vtss_phy_ts_stats_t::ingr_pream_shrink_err
```

Frames with preambles too short to shrink

Definition at line 1575 of file vtss_phy_ts_api.h.

7.201.2.2 egr_pream_shrink_err

```
u32 vtss_phy_ts_stats_t::egr_pream_shrink_err
```

Frames with preambles too short to shrink

Definition at line 1576 of file vtss_phy_ts_api.h.

7.201.2.3 ingr_fcs_err

```
u32 vtss_phy_ts_stats_t::ingr_fcs_err
```

Timestamp block received frame with FCS error in ingress

Definition at line 1577 of file vtss_phy_ts_api.h.

7.201.2.4 egr_fcs_err

`u32 vtss_phy_ts_stats_t::egr_fcs_err`

Timestamp block received frame with FCS error in egress

Definition at line 1578 of file `vtss_phy_ts_api.h`.

7.201.2.5 ingr_frm_mod_cnt

`u32 vtss_phy_ts_stats_t::ingr_frm_mod_cnt`

No of frames modified by timestamp block (rewritter) in ingress

Definition at line 1579 of file `vtss_phy_ts_api.h`.

7.201.2.6 egr_frm_mod_cnt

`u32 vtss_phy_ts_stats_t::egr_frm_mod_cnt`

No of frames modified by timestamp block (rewritter) in egress

Definition at line 1580 of file `vtss_phy_ts_api.h`.

7.201.2.7 ts_fifo_tx_cnt

`u32 vtss_phy_ts_stats_t::ts_fifo_tx_cnt`

the number of timestamps transmitted to the interface

Definition at line 1581 of file `vtss_phy_ts_api.h`.

7.201.2.8 ts_fifo_drop_cnt

`u32 vtss_phy_ts_stats_t::ts_fifo_drop_cnt`

Count of dropped Timestamps not enqueued to the Tx TSFIFO

Definition at line 1582 of file `vtss_phy_ts_api.h`.

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

- `vtss_api/include/vtss_phy_ts_api.h`

7.202 vtss_phy_ts_y1731_oam_conf_t Struct Reference

Analyzer OAM comparator, Y.1731 OAM Packet format configuration options.

```
#include <vtss_phy_ts_api.h>
```

Data Fields

- `BOOL range_en`
- `vtss_phy_ts_y1731_oam_delaym_type_t delaym_type`
- `union {`
 - `struct {`
 - `u8 val`
 - `u8 mask`
 - `} value`
 - `struct {`
 - `u8 upper`
 - `u8 lower`
 - `} range`
- `} meg_level`

7.202.1 Detailed Description

Analyzer OAM comparator, Y.1731 OAM Packet format configuration options.

Definition at line 1342 of file vtss_phy_ts_api.h.

7.202.2 Field Documentation

7.202.2.1 range_en

```
BOOL vtss_phy_ts_y1731_oam_conf_t::range_en
```

OAM MEG level in range enable/disable

Definition at line 1343 of file vtss_phy_ts_api.h.

7.202.2.2 delaym_type

```
vtss_phy_ts_y1731_oam_delaym_type_t vtss_phy_ts_y1731_oam_conf_t::delaym_type
```

OAM delay measurement method

Definition at line 1344 of file vtss_phy_ts_api.h.

7.202.2.3 val

`u8 vtss_phy_ts_y1731_oam_conf_t::val`

Value

Definition at line 1347 of file vtss_phy_ts_api.h.

7.202.2.4 mask

`u8 vtss_phy_ts_y1731_oam_conf_t::mask`

Mask

Definition at line 1348 of file vtss_phy_ts_api.h.

7.202.2.5 value

`struct { ... } vtss_phy_ts_y1731_oam_conf_t::value`

specific MEG Level, for don't care set mask as '0'

7.202.2.6 upper

`u8 vtss_phy_ts_y1731_oam_conf_t::upper`

Range Upper value

Definition at line 1351 of file vtss_phy_ts_api.h.

7.202.2.7 lower

`u8 vtss_phy_ts_y1731_oam_conf_t::lower`

Range lower value

Definition at line 1352 of file vtss_phy_ts_api.h.

7.202.2.8 range

`struct { ... } vtss_phy_ts_y1731_oam_conf_t::range`

Range configuration

7.202.2.9 meg_level

```
union { ... } vtss_phy_ts_y1731_oam_conf_t::meg_level
```

OAM MEG level config

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

- [vtss_api/include/vtss_phy_ts_api.h](#)

7.203 vtss_phy_type_t Struct Reference

Phy type information.

```
#include <vtss_phy_api.h>
```

Data Fields

- [u16 part_number](#)
- [u16 revision](#)
- [u8 port_cnt](#)
- [u16 channel_id](#)
- [u16 base_port_no](#)
- [vtss_port_no_t phy_api_base_no](#)

7.203.1 Detailed Description

Phy type information.

Definition at line 143 of file [vtss_phy_api.h](#).

7.203.2 Field Documentation

7.203.2.1 part_number

```
u16 vtss_phy_type_t::part_number
```

Part number

Definition at line 145 of file [vtss_phy_api.h](#).

7.203.2.2 revision

`u16 vtss_phy_type_t::revision`

Chip revision

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

7.203.2.3 port_cnt

`u8 vtss_phy_type_t::port_cnt`

The number of PHY ports in the chip

Definition at line 147 of file `vtss_phy_api.h`.

7.203.2.4 channel_id

`u16 vtss_phy_type_t::channel_id`

Channel id

Definition at line 148 of file `vtss_phy_api.h`.

7.203.2.5 base_port_no

`u16 vtss_phy_type_t::base_port_no`

The port number for the first PHY port within the chip.

Definition at line 149 of file `vtss_phy_api.h`.

7.203.2.6 phy_api_base_no

`vtss_port_no_t vtss_phy_type_t::phy_api_base_no`

First API no within this phy (in' case of multiple channels)

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

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

- `vtss_api/include/vtss_phy_api.h`

7.204 vtss_phy_veriphy_result_t Struct Reference

VeriPHY result.

```
#include <vtss_phy_api.h>
```

Data Fields

- `BOOL link`
- `vtss_phy_veriphy_status_t status [4]`
- `u8 length [4]`

7.204.1 Detailed Description

VeriPHY result.

Definition at line 958 of file vtss_phy_api.h.

7.204.2 Field Documentation

7.204.2.1 link

```
BOOL vtss_phy_veriphy_result_t::link
```

Link status

Definition at line 959 of file vtss_phy_api.h.

7.204.2.2 status

```
vtss_phy_veriphy_status_t vtss_phy_veriphy_result_t::status[4]
```

Status, pair A-D (0-3)

Definition at line 960 of file vtss_phy_api.h.

7.204.2.3 length

```
u8 vtss_phy_veriphy_result_t::length[4]
```

Length (meters), pair A-D (0-3)

Definition at line 961 of file vtss_phy_api.h.

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

- vtss_api/include/[vtss_phy_api.h](#)

7.205 vtss_phy_wol_conf_t Struct Reference

Structure for Get/Set Wake-On-LAN configuration.

```
#include <vtss_phy_api.h>
```

Data Fields

- `BOOL secure_on_enable`
- `vtss_wol_mac_addr_t wol_mac`
- `vtss_secure_on_passwd_t wol_pass`
- `vtss_wol_passwd_len_type_t wol_passwd_len`
- `u16 magic_pkt_cnt`

7.205.1 Detailed Description

Structure for Get/Set Wake-On-LAN configuration.

Definition at line 1680 of file vtss_phy_api.h.

7.205.2 Field Documentation

7.205.2.1 secure_on_enable

```
BOOL vtss_phy_wol_conf_t::secure_on_enable
```

Enable/Disable for Secure-On Password

Definition at line 1681 of file vtss_phy_api.h.

7.205.2.2 wol_mac

```
vtss_wol_mac_addr_t vtss_phy_wol_conf_t::wol_mac
```

Wake-On-LAN MAC Addr Definition

Definition at line 1682 of file vtss_phy_api.h.

7.205.2.3 wol_pass

```
vtss_secure_on_passwd_t vtss_phy_wol_conf_t::wol_pass
```

Wake-On-LAN Password Definition

Definition at line 1683 of file vtss_phy_api.h.

7.205.2.4 wol_passwd_len

```
vtss_wol_passwd_len_type_t vtss_phy_wol_conf_t::wol_passwd_len
```

Enumeration for Password Length options

Definition at line 1684 of file vtss_phy_api.h.

7.205.2.5 magic_pkt_cnt

```
u16 vtss_phy_wol_conf_t::magic_pkt_cnt
```

Magic Packet Repetition Count (1-16 is valid)

Definition at line 1685 of file vtss_phy_api.h.

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

- vtss_api/include/[vtss_phy_api.h](#)

7.206 vtss_pi_conf_t Struct Reference

PI configuration.

```
#include <vtss_init_api.h>
```

Data Fields

- `u32 cs_wait_ns`

7.206.1 Detailed Description

PI configuration.

Definition at line 322 of file `vtss_init_api.h`.

7.206.2 Field Documentation

7.206.2.1 `cs_wait_ns`

`u32 vtss_pi_conf_t::cs_wait_ns`

Minimum CS wait time in nanoseconds

Definition at line 327 of file `vtss_init_api.h`.

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

- `vtss_api/include/vtss_init_api.h`

7.207 `vtss_policer_ext_t` Struct Reference

Policer Extensions.

```
#include <vtss_qos_api.h>
```

Data Fields

- `BOOL frame_rate`
- `vtss_dp_level_t dp_bypass_level`
- `BOOL unicast`
- `BOOL multicast`
- `BOOL broadcast`
- `BOOL uc_no_flood`
- `BOOL mc_no_flood`
- `BOOL flooded`
- `BOOL learning`
- `BOOL to_cpu`
- `BOOL cpu_queue [VTSS_PORT_POLICER_CPU_QUEUES]`
- `BOOL limit_noncpu_traffic`
- `BOOL limit_cpu_traffic`
- `BOOL flow_control`

7.207.1 Detailed Description

Policer Extensions.

Definition at line 198 of file vtss_qos_api.h.

7.207.2 Field Documentation

7.207.2.1 frame_rate

`BOOL vtss_policer_ext_t::frame_rate`

Measure rates in frames per seconds instead of bits per second

Definition at line 201 of file vtss_qos_api.h.

7.207.2.2 dp_bypass_level

`vtss_dp_level_t vtss_policer_ext_t::dp_bypass_level`

Drop Predence bypass level

Definition at line 204 of file vtss_qos_api.h.

7.207.2.3 unicast

`BOOL vtss_policer_ext_t::unicast`

Unicast frames are policed

Definition at line 207 of file vtss_qos_api.h.

7.207.2.4 multicast

`BOOL vtss_policer_ext_t::multicast`

Multicast frames are policed

Definition at line 208 of file vtss_qos_api.h.

7.207.2.5 broadcast

`BOOL vtss_policer_ext_t::broadcast`

Broadcast frames are policed

Definition at line 209 of file `vtss_qos_api.h`.

7.207.2.6 uc_no_flood

`BOOL vtss_policer_ext_t::uc_no_flood`

Exclude flooding unicast frames (if unicast is set)

Definition at line 210 of file `vtss_qos_api.h`.

7.207.2.7 mc_no_flood

`BOOL vtss_policer_ext_t::mc_no_flood`

Exclude flooding multicast frames (if multicast is set)

Definition at line 211 of file `vtss_qos_api.h`.

7.207.2.8 flooded

`BOOL vtss_policer_ext_t::flooded`

Flooded frames are policed

Definition at line 212 of file `vtss_qos_api.h`.

7.207.2.9 learning

`BOOL vtss_policer_ext_t::learning`

Learning frames are policed

Definition at line 223 of file `vtss_qos_api.h`.

7.207.2.10 to_cpu

`BOOL vtss_policer_ext_t::to_cpu`

Frames to the CPU are policed

Definition at line 224 of file vtss_qos_api.h.

7.207.2.11 cpu_queue

`BOOL vtss_policer_ext_t::cpu_queue[VTSS_PORT_POLICER_CPU_QUEUES]`

Enable each individual CPU queue (if to_cpu is set)

Definition at line 225 of file vtss_qos_api.h.

7.207.2.12 limit_noncpu_traffic

`BOOL vtss_policer_ext_t::limit_noncpu_traffic`

Remove the front ports from the destination set for a policed frame

Definition at line 226 of file vtss_qos_api.h.

7.207.2.13 limit_cpu_traffic

`BOOL vtss_policer_ext_t::limit_cpu_traffic`

Remove the CPU ports from the destination set for a policed frame

Definition at line 227 of file vtss_qos_api.h.

7.207.2.14 flow_control

`BOOL vtss_policer_ext_t::flow_control`

Flow control is enabled

Definition at line 230 of file vtss_qos_api.h.

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

- vtss_api/include/[vtss_qos_api.h](#)

7.208 vtss_policer_t Struct Reference

Policer.

```
#include <vtss_qos_api.h>
```

Data Fields

- `vtss_burst_level_t level`
- `vtss_bitrate_t rate`

7.208.1 Detailed Description

Policer.

Definition at line 185 of file `vtss_qos_api.h`.

7.208.2 Field Documentation

7.208.2.1 level

```
vtss_burst_level_t vtss_policer_t::level
```

Burst level

Definition at line 187 of file `vtss_qos_api.h`.

7.208.2.2 rate

```
vtss_bitrate_t vtss_policer_t::rate
```

Maximum rate

Definition at line 188 of file `vtss_qos_api.h`.

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

- `vtss_api/include/vtss_qos_api.h`

7.209 vtss_port_bridge_counters_t Struct Reference

Port bridge counter structure (RFC 4188)

```
#include <port.h>
```

Data Fields

- [vtss_port_counter_t dot1dTpPortInDiscards](#)

7.209.1 Detailed Description

Port bridge counter structure (RFC 4188)

Definition at line 201 of file port.h.

7.209.2 Field Documentation

7.209.2.1 dot1dTpPortInDiscards

[vtss_port_counter_t](#) vtss_port_bridge_counters_t::dot1dTpPortInDiscards

Rx bridge discards

Definition at line 203 of file port.h.

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

- [vtss_api/include/vtss/api/port.h](#)

7.210 vtss_port_clause_37_adv_t Struct Reference

Advertisement control data for Clause 37 aneg.

```
#include <vtss_port_api.h>
```

Data Fields

- [BOOL fdx](#)
- [BOOL hdx](#)
- [BOOL symmetric_pause](#)
- [BOOL asymmetric_pause](#)
- [vtss_port_clause_37_remote_fault_t remote_fault](#)
- [BOOL acknowledge](#)
- [BOOL next_page](#)

7.210.1 Detailed Description

Advertisement control data for Clause 37 aneg.

Definition at line 127 of file vtss_port_api.h.

7.210.2 Field Documentation

7.210.2.1 fdx

`BOOL vtss_port_clause_37_adv_t::fdx`

(FD)

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

7.210.2.2 hdx

`BOOL vtss_port_clause_37_adv_t::hdx`

(HD)

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

7.210.2.3 symmetric_pause

`BOOL vtss_port_clause_37_adv_t::symmetric_pause`

(PS1)

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

7.210.2.4 asymmetric_pause

`BOOL vtss_port_clause_37_adv_t::asymmetric_pause`

(PS2)

Definition at line 132 of file `vtss_port_api.h`.

7.210.2.5 remote_fault

`vtss_port_clause_37_remote_fault_t vtss_port_clause_37_adv_t::remote_fault`

(RF1) + (RF2)

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

7.210.2.6 acknowledge

`BOOL vtss_port_clause_37_adv_t::acknowledge`

(Ack)

Definition at line 134 of file vtss_port_api.h.

7.210.2.7 next_page

`BOOL vtss_port_clause_37_adv_t::next_page`

(NP)

Definition at line 135 of file vtss_port_api.h.

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

- `vtss_api/include/vtss_port_api.h`

7.211 vtss_port_clause_37_control_t Struct Reference

Auto-negotiation control parameter struct.

```
#include <vtss_port_api.h>
```

Data Fields

- `BOOL enable`
- `vtss_port_clause_37_adv_t advertisement`

7.211.1 Detailed Description

Auto-negotiation control parameter struct.

Definition at line 152 of file vtss_port_api.h.

7.211.2 Field Documentation

7.211.2.1 enable

`BOOL vtss_port_clause_37_control_t::enable`

Enable of Autoneg

Definition at line 154 of file `vtss_port_api.h`.

7.211.2.2 advertisement

`vtss_port_clause_37_adv_t vtss_port_clause_37_control_t::advertisement`

Clause 37 Advertisement control data

Definition at line 155 of file `vtss_port_api.h`.

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

- `vtss_api/include/vtss_port_api.h`

7.212 `vtss_port_conf_t` Struct Reference

Port configuration structure.

```
#include <vtss_port_api.h>
```

Data Fields

- `vtss_port_interface_t if_type`
- `BOOL sd_enable`
- `BOOL sd_active_high`
- `BOOL sd_internal`
- `vtss_port_frame_gaps_t frame_gaps`
- `BOOL power_down`
- `vtss_port_speed_t speed`
- `BOOL fdx`
- `vtss_port_flow_control_conf_t flow_control`
- `u32 max_frame_length`
- `BOOL frame_length_chk`
- `vtss_port_max_tags_t max_tags`
- `BOOL exc_col_cont`
- `BOOL xaui_rx_lane_flip`
- `BOOL xaui_tx_lane_flip`
- `vtss_port_loop_t loop`
- `vtss_port_serdes_conf_t serdes`

7.212.1 Detailed Description

Port configuration structure.

Definition at line 268 of file vtss_port_api.h.

7.212.2 Field Documentation

7.212.2.1 `if_type`

`vtss_port_interface_t` `vtss_port_conf_t::if_type`

Interface type

Definition at line 270 of file vtss_port_api.h.

7.212.2.2 `sd_enable`

`BOOL` `vtss_port_conf_t::sd_enable`

Signal detect enable

Definition at line 271 of file vtss_port_api.h.

7.212.2.3 `sd_active_high`

`BOOL` `vtss_port_conf_t::sd_active_high`

External signal detect polarity

Definition at line 272 of file vtss_port_api.h.

7.212.2.4 `sd_internal`

`BOOL` `vtss_port_conf_t::sd_internal`

Internal signal detect selection

Definition at line 273 of file vtss_port_api.h.

7.212.2.5 frame_gaps

`vtss_port_frame_gaps_t` `vtss_port_conf_t::frame_gaps`

Inter frame gaps

Definition at line 274 of file vtss_port_api.h.

7.212.2.6 power_down

`BOOL` `vtss_port_conf_t::power_down`

Disable and power down the port

Definition at line 275 of file vtss_port_api.h.

7.212.2.7 speed

`vtss_port_speed_t` `vtss_port_conf_t::speed`

Port speed

Definition at line 276 of file vtss_port_api.h.

7.212.2.8 fdx

`BOOL` `vtss_port_conf_t::fdx`

Full duplex mode

Definition at line 277 of file vtss_port_api.h.

7.212.2.9 flow_control

`vtss_port_flow_control_conf_t` `vtss_port_conf_t::flow_control`

Flow control setup

Definition at line 278 of file vtss_port_api.h.

7.212.2.10 max_frame_length

`u32 vtss_port_conf_t::max_frame_length`

Maximum frame length

Definition at line 279 of file vtss_port_api.h.

7.212.2.11 frame_length_chk

`BOOL vtss_port_conf_t::frame_length_chk`

Enforce 802.3 frame length check (from ethertype field)

Definition at line 280 of file vtss_port_api.h.

7.212.2.12 max_tags

`vtss_port_max_tags_t vtss_port_conf_t::max_tags`

VLAN awareness for length check

Definition at line 281 of file vtss_port_api.h.

7.212.2.13 exc_col_cont

`BOOL vtss_port_conf_t::exc_col_cont`

Excessive collision continuation

Definition at line 282 of file vtss_port_api.h.

7.212.2.14 xaui_rx_lane_flip

`BOOL vtss_port_conf_t::xaui_rx_lane_flip`

Xaui Rx lane flip

Definition at line 283 of file vtss_port_api.h.

7.212.2.15 xauि_tx_lane_flip

`BOOL vtss_port_conf_t::xauि_tx_lane_flip`

Xauि Tx lane flip

Definition at line 284 of file vtss_port_api.h.

7.212.2.16 loop

`vtss_port_loop_t vtss_port_conf_t::loop`

Enable/disable of port loop back

Definition at line 285 of file vtss_port_api.h.

7.212.2.17 serdes

`vtss_port_serdes_conf_t vtss_port_conf_t::serdes`

Serdes settings (for SFI interface)

Definition at line 286 of file vtss_port_api.h.

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

- vtss_api/include/[vtss_port_api.h](#)

7.213 vtss_port_counters_t Struct Reference

Port counter structure.

```
#include <port.h>
```

Data Fields

- [vtss_port_rmon_counters_t rmon](#)
- [vtss_port_if_group_counters_t if_group](#)
- [vtss_port_ethernet_like_counters_t ethernet_like](#)
- [vtss_port_bridge_counters_t bridge](#)
- [vtss_port_proprietary_counters_t prop](#)

7.213.1 Detailed Description

Port counter structure.

Definition at line 220 of file port.h.

7.213.2 Field Documentation

7.213.2.1 rmon

```
vtss_port_rmon_counters_t vtss_port_counters_t::rmon
```

RMON counters

Definition at line 222 of file port.h.

7.213.2.2 if_group

```
vtss_port_if_group_counters_t vtss_port_counters_t::if_group
```

Interfaces Group counters

Definition at line 223 of file port.h.

7.213.2.3 ethernet_like

```
vtss_port_ethernet_like_counters_t vtss_port_counters_t::ethernet_like
```

Ethernet-like Interface counters

Definition at line 224 of file port.h.

7.213.2.4 bridge

```
vtss_port_bridge_counters_t vtss_port_counters_t::bridge
```

Bridge counters

Definition at line 227 of file port.h.

7.213.2.5 prop

```
vtss_port_proprietary_counters_t vtss_port_counters_t::prop
```

Proprietary counters

Definition at line 230 of file port.h.

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

- [vtss_api/include/vtss/api/port.h](#)

7.214 vtss_port_ethernet_like_counters_t Struct Reference

Ethernet-like Interface counter structure (RFC 3635)

```
#include <port.h>
```

Data Fields

- [vtss_port_counter_t dot3StatsAlignmentErrors](#)
- [vtss_port_counter_t dot3StatsFCSErrors](#)
- [vtss_port_counter_t dot3StatsFrameTooLongs](#)
- [vtss_port_counter_t dot3StatsSymbolErrors](#)
- [vtss_port_counter_t dot3ControlInUnknownOpcodes](#)
- [vtss_port_counter_t dot3InPauseFrames](#)
- [vtss_port_counter_t dot3StatsSingleCollisionFrames](#)
- [vtss_port_counter_t dot3StatsMultipleCollisionFrames](#)
- [vtss_port_counter_t dot3StatsDeferredTransmissions](#)
- [vtss_port_counter_t dot3StatsLateCollisions](#)
- [vtss_port_counter_t dot3StatsExcessiveCollisions](#)
- [vtss_port_counter_t dot3StatsCarrierSenseErrors](#)
- [vtss_port_counter_t dot3OutPauseFrames](#)

7.214.1 Detailed Description

Ethernet-like Interface counter structure (RFC 3635)

Definition at line 163 of file port.h.

7.214.2 Field Documentation

7.214.2.1 dot3StatsAlignmentErrors

`vtss_port_counter_t vtss_port_ethernet_like_counters_t::dot3StatsAlignmentErrors`

Rx alignment errors

Definition at line 165 of file port.h.

7.214.2.2 dot3StatsFCSErrors

`vtss_port_counter_t vtss_port_ethernet_like_counters_t::dot3StatsFCSErrors`

Rx FCS errors

Definition at line 166 of file port.h.

7.214.2.3 dot3StatsFrameTooLongs

`vtss_port_counter_t vtss_port_ethernet_like_counters_t::dot3StatsFrameTooLongs`

Rx too long

Definition at line 167 of file port.h.

7.214.2.4 dot3StatsSymbolErrors

`vtss_port_counter_t vtss_port_ethernet_like_counters_t::dot3StatsSymbolErrors`

Rx symbol errors

Definition at line 168 of file port.h.

7.214.2.5 dot3ControlInUnknownOpcodes

`vtss_port_counter_t vtss_port_ethernet_like_counters_t::dot3ControlInUnknownOpcodes`

Rx unknown opcodes

Definition at line 169 of file port.h.

7.214.2.6 dot3InPauseFrames

`vtss_port_counter_t vtss_port_ethernet_like_counters_t::dot3InPauseFrames`

Rx pause

Definition at line 171 of file port.h.

7.214.2.7 dot3StatsSingleCollisionFrames

`vtss_port_counter_t vtss_port_ethernet_like_counters_t::dot3StatsSingleCollisionFrames`

Tx single collisions

Definition at line 174 of file port.h.

7.214.2.8 dot3StatsMultipleCollisionFrames

`vtss_port_counter_t vtss_port_ethernet_like_counters_t::dot3StatsMultipleCollisionFrames`

Tx multiple collisions

Definition at line 175 of file port.h.

7.214.2.9 dot3StatsDeferredTransmissions

`vtss_port_counter_t vtss_port_ethernet_like_counters_t::dot3StatsDeferredTransmissions`

Tx deferred

Definition at line 176 of file port.h.

7.214.2.10 dot3StatsLateCollisions

`vtss_port_counter_t vtss_port_ethernet_like_counters_t::dot3StatsLateCollisions`

Tx late collisions

Definition at line 177 of file port.h.

7.214.2.11 dot3StatsExcessiveCollisions

`vtss_port_counter_t vtss_port_ethernet_like_counters_t::dot3StatsExcessiveCollisions`

Tx excessive collisions

Definition at line 178 of file port.h.

7.214.2.12 dot3StatsCarrierSenseErrors

`vtss_port_counter_t vtss_port_ethernet_like_counters_t::dot3StatsCarrierSenseErrors`

Tx carrier sense errors

Definition at line 179 of file port.h.

7.214.2.13 dot3OutPauseFrames

`vtss_port_counter_t vtss_port_ethernet_like_counters_t::dot3OutPauseFrames`

Tx pause

Definition at line 181 of file port.h.

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

- `vtss_api/include/vtss/api/port.h`

7.215 vtss_port_flow_control_conf_t Struct Reference

Flow control setup.

```
#include <vtss_port_api.h>
```

Data Fields

- `BOOL obey`
- `BOOL generate`
- `vtss_mac_t smac`
- `BOOL pfc [VTSS_PRIOS]`

7.215.1 Detailed Description

Flow control setup.

Definition at line 189 of file vtss_port_api.h.

7.215.2 Field Documentation

7.215.2.1 obey

`BOOL vtss_port_flow_control_conf_t::obey`

TRUE if 802.3x PAUSE frames should be obeyed

Definition at line 191 of file vtss_port_api.h.

7.215.2.2 generate

`BOOL vtss_port_flow_control_conf_t::generate`

TRUE if 802.3x PAUSE frames should be generated

Definition at line 195 of file vtss_port_api.h.

7.215.2.3 smac

`vtss_mac_t vtss_port_flow_control_conf_t::smac`

Port MAC address used as SMAC in PAUSE frames

Definition at line 196 of file vtss_port_api.h.

7.215.2.4 pfc

`BOOL vtss_port_flow_control_conf_t::pfc[VTSS_PRIOS]`

TRUE if 802.1Qbb Priority Flow Control should be generated and obeyed. Cannot be enabled together with 802.3x Flowcontrol

Definition at line 198 of file vtss_port_api.h.

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

- vtss_api/include/vtss_port_api.h

7.216 vtss_port_frame_gaps_t Struct Reference

Inter frame gap structure.

```
#include <vtss_port_api.h>
```

Data Fields

- `u32 hdx_gap_1`
- `u32 hdx_gap_2`
- `u32 fdx_gap`

7.216.1 Detailed Description

Inter frame gap structure.

Definition at line 206 of file vtss_port_api.h.

7.216.2 Field Documentation

7.216.2.1 hdx_gap_1

```
u32 vtss_port_frame_gaps_t::hdx_gap_1
```

Half duplex: First part of Rx to Tx gap

Definition at line 208 of file vtss_port_api.h.

7.216.2.2 hdx_gap_2

```
u32 vtss_port_frame_gaps_t::hdx_gap_2
```

Half duplex: Second part of Rx to Tx gap

Definition at line 209 of file vtss_port_api.h.

7.216.2.3 fdx_gap

```
u32 vtss_port_frame_gaps_t::fdx_gap
```

Full duplex: Tx to Tx gap

Definition at line 210 of file vtss_port_api.h.

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

- [vtss_api/include/vtss_port_api.h](#)

7.217 vtss_port_if_group_counters_t Struct Reference

Interfaces Group counter structure (RFC 2863)

```
#include <port.h>
```

Data Fields

- [vtss_port_counter_t ifInOctets](#)
- [vtss_port_counter_t ifInUcastPkts](#)
- [vtss_port_counter_t ifInMulticastPkts](#)
- [vtss_port_counter_t ifInBroadcastPkts](#)
- [vtss_port_counter_t ifInNUcastPkts](#)
- [vtss_port_counter_t ifInDiscards](#)
- [vtss_port_counter_t ifInErrors](#)
- [vtss_port_counter_t ifOutOctets](#)
- [vtss_port_counter_t ifOutUcastPkts](#)
- [vtss_port_counter_t ifOutMulticastPkts](#)
- [vtss_port_counter_t ifOutBroadcastPkts](#)
- [vtss_port_counter_t ifOutNUcastPkts](#)
- [vtss_port_counter_t ifOutDiscards](#)
- [vtss_port_counter_t ifOutErrors](#)

7.217.1 Detailed Description

Interfaces Group counter structure (RFC 2863)

Definition at line 144 of file port.h.

7.217.2 Field Documentation

7.217.2.1 ifInOctets

`vtss_port_counter_t vtss_port_if_group_counters_t::ifInOctets`

Rx octets

Definition at line 145 of file port.h.

7.217.2.2 ifInUcastPkts

`vtss_port_counter_t vtss_port_if_group_counters_t::ifInUcastPkts`

Rx unicasts

Definition at line 146 of file port.h.

7.217.2.3 ifInMulticastPkts

`vtss_port_counter_t vtss_port_if_group_counters_t::ifInMulticastPkts`

Rx multicasts

Definition at line 147 of file port.h.

7.217.2.4 ifInBroadcastPkts

`vtss_port_counter_t vtss_port_if_group_counters_t::ifInBroadcastPkts`

Rx broadcasts

Definition at line 148 of file port.h.

7.217.2.5 ifInNUcastPkts

`vtss_port_counter_t vtss_port_if_group_counters_t::ifInNUcastPkts`

Rx non-unicasts

Definition at line 149 of file port.h.

7.217.2.6 ifInDiscards

```
vtss_port_counter_t vtss_port_if_group_counters_t::ifInDiscards
```

Rx discards

Definition at line 150 of file port.h.

7.217.2.7 ifInErrors

```
vtss_port_counter_t vtss_port_if_group_counters_t::ifInErrors
```

Rx errors

Definition at line 151 of file port.h.

7.217.2.8 ifOutOctets

```
vtss_port_counter_t vtss_port_if_group_counters_t::ifOutOctets
```

Tx octets

Definition at line 153 of file port.h.

7.217.2.9 ifOutUcastPkts

```
vtss_port_counter_t vtss_port_if_group_counters_t::ifOutUcastPkts
```

Tx unicasts

Definition at line 154 of file port.h.

7.217.2.10 ifOutMulticastPkts

```
vtss_port_counter_t vtss_port_if_group_counters_t::ifOutMulticastPkts
```

Tx multicasts

Definition at line 155 of file port.h.

7.217.2.11 ifOutBroadcastPkts

```
vtss_port_counter_t vtss_port_if_group_counters_t::ifOutBroadcastPkts
```

Tx broadcasts

Definition at line 156 of file port.h.

7.217.2.12 ifOutNUcastPkts

```
vtss_port_counter_t vtss_port_if_group_counters_t::ifOutNUcastPkts
```

Tx non-unicasts

Definition at line 157 of file port.h.

7.217.2.13 ifOutDiscards

```
vtss_port_counter_t vtss_port_if_group_counters_t::ifOutDiscards
```

Tx discards

Definition at line 158 of file port.h.

7.217.2.14 ifOutErrors

```
vtss_port_counter_t vtss_port_if_group_counters_t::ifOutErrors
```

Tx errors

Definition at line 159 of file port.h.

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

- [vtss_api/include/vtss/api/port.h](#)

7.218 **vtss_port_ifh_t** Struct Reference

Port Internal Frame Header structure.

```
#include <vtss_port_api.h>
```

Data Fields

- `BOOL ena_ifh_header`

7.218.1 Detailed Description

Port Internal Frame Header structure.

Definition at line 434 of file `vtss_port_api.h`.

7.218.2 Field Documentation

7.218.2.1 `ena_ifh_header`

`BOOL vtss_port_ifh_t::ena_ifh_header`

At egress keep IFH

Definition at line 450 of file `vtss_port_api.h`.

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

- `vtss_api/include/vtss_port_api.h`

7.219 `vtss_port_map_t` Struct Reference

Port map structure.

```
#include <vtss_port_api.h>
```

Data Fields

- `i32 chip_port`
- `vtss_chip_no_t chip_no`
- `vtss_miim_controller_t miim_controller`
- `u8 miim_addr`
- `vtss_chip_no_t miim_chip_no`

7.219.1 Detailed Description

Port map structure.

Definition at line 72 of file `vtss_port_api.h`.

7.219.2 Field Documentation

7.219.2.1 chip_port

`i32 vtss_port_map_t::chip_port`

Set to -1 if not used

Definition at line 74 of file vtss_port_api.h.

7.219.2.2 chip_no

`vtss_chip_no_t vtss_port_map_t::chip_no`

Chip number, multi-chip targets

Definition at line 75 of file vtss_port_api.h.

7.219.2.3 miim_controller

`vtss_miim_controller_t vtss_port_map_t::miim_controller`

MII management controller

Definition at line 80 of file vtss_port_api.h.

7.219.2.4 miim_addr

`u8 vtss_port_map_t::miim_addr`

PHY address, ignored for VTSS_MIIM_CONTROLLER_NONE

Definition at line 81 of file vtss_port_api.h.

7.219.2.5 miim_chip_no

`vtss_chip_no_t vtss_port_map_t::miim_chip_no`

MII management chip number, multi-chip targets

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

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

- `vtss_api/include/vtss_port_api.h`

7.220 vtss_port_proprietary_counters_t Struct Reference

Port proprietary counter structure.

```
#include <port.h>
```

Data Fields

- `vtss_port_counter_t rx_prio [VTSS_PRIOS]`
- `vtss_port_counter_t tx_prio [VTSS_PRIOS]`

7.220.1 Detailed Description

Port proprietary counter structure.

Definition at line 207 of file `port.h`.

7.220.2 Field Documentation

7.220.2.1 rx_prio

`vtss_port_counter_t vtss_port_proprietary_counters_t::rx_prio[VTSS_PRIOS]`

Rx frames

Definition at line 210 of file `port.h`.

7.220.2.2 tx_prio

```
vtss_port_counter_t vtss_port_proprietary_counters_t::tx_prio[VTSS_PRIOS]
```

Tx frames

Definition at line 214 of file port.h.

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

- vtss_api/include/vtss/api/port.h

7.221 vtss_port_rmon_counters_t Struct Reference

RMON counter structure (RFC 2819)

```
#include <port.h>
```

Data Fields

- vtss_port_counter_t rx_etherStatsDropEvents
- vtss_port_counter_t rx_etherStatsOctets
- vtss_port_counter_t rx_etherStatsPkts
- vtss_port_counter_t rx_etherStatsBroadcastPkts
- vtss_port_counter_t rx_etherStatsMulticastPkts
- vtss_port_counter_t rx_etherStatsCRCAlignErrors
- vtss_port_counter_t rx_etherStatsUndersizePkts
- vtss_port_counter_t rx_etherStatsOversizePkts
- vtss_port_counter_t rx_etherStatsFragments
- vtss_port_counter_t rx_etherStatsJabbers
- vtss_port_counter_t rx_etherStatsPkts64Octets
- vtss_port_counter_t rx_etherStatsPkts65to127Octets
- vtss_port_counter_t rx_etherStatsPkts128to255Octets
- vtss_port_counter_t rx_etherStatsPkts256to511Octets
- vtss_port_counter_t rx_etherStatsPkts512to1023Octets
- vtss_port_counter_t rx_etherStatsPkts1024to1518Octets
- vtss_port_counter_t rx_etherStatsPkts1519toMaxOctets
- vtss_port_counter_t tx_etherStatsDropEvents
- vtss_port_counter_t tx_etherStatsOctets
- vtss_port_counter_t tx_etherStatsPkts
- vtss_port_counter_t tx_etherStatsBroadcastPkts
- vtss_port_counter_t tx_etherStatsMulticastPkts
- vtss_port_counter_t tx_etherStatsCollisions
- vtss_port_counter_t tx_etherStatsPkts64Octets
- vtss_port_counter_t tx_etherStatsPkts65to127Octets
- vtss_port_counter_t tx_etherStatsPkts128to255Octets
- vtss_port_counter_t tx_etherStatsPkts256to511Octets
- vtss_port_counter_t tx_etherStatsPkts512to1023Octets
- vtss_port_counter_t tx_etherStatsPkts1024to1518Octets
- vtss_port_counter_t tx_etherStatsPkts1519toMaxOctets

7.221.1 Detailed Description

RMON counter structure (RFC 2819)

Definition at line 109 of file port.h.

7.221.2 Field Documentation

7.221.2.1 rx_etherStatsDropEvents

```
vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsDropEvents
```

Rx drop events

Definition at line 110 of file port.h.

7.221.2.2 rx_etherStatsOctets

```
vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsOctets
```

Rx octets

Definition at line 111 of file port.h.

7.221.2.3 rx_etherStatsPkts

```
vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsPkts
```

Rx packets

Definition at line 112 of file port.h.

7.221.2.4 rx_etherStatsBroadcastPkts

```
vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsBroadcastPkts
```

Rx broadcasts

Definition at line 113 of file port.h.

7.221.2.5 rx_etherStatsMulticastPkts

```
vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsMulticastPkts
```

Rx multicasts

Definition at line 114 of file port.h.

7.221.2.6 rx_etherStatsCRCAlignErrors

```
vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsCRCAlignErrors
```

Rx CRC/alignment errors

Definition at line 115 of file port.h.

7.221.2.7 rx_etherStatsUndersizePkts

```
vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsUndersizePkts
```

Rx undersize packets

Definition at line 116 of file port.h.

7.221.2.8 rx_etherStatsOversizePkts

```
vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsOversizePkts
```

Rx oversize packets

Definition at line 117 of file port.h.

7.221.2.9 rx_etherStatsFragments

```
vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsFragments
```

Rx fragments

Definition at line 118 of file port.h.

7.221.2.10 rx_etherStatsJabbers

```
vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsJabbers
```

Rx jabbers

Definition at line 119 of file port.h.

7.221.2.11 rx_etherStatsPkts64Octets

```
vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsPkts64Octets
```

Rx 64 byte packets

Definition at line 120 of file port.h.

7.221.2.12 rx_etherStatsPkts65to127Octets

```
vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsPkts65to127Octets
```

Rx 65-127 byte packets

Definition at line 121 of file port.h.

7.221.2.13 rx_etherStatsPkts128to255Octets

```
vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsPkts128to255Octets
```

Rx 128-255 byte packets

Definition at line 122 of file port.h.

7.221.2.14 rx_etherStatsPkts256to511Octets

```
vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsPkts256to511Octets
```

Rx 256-511 byte packets

Definition at line 123 of file port.h.

7.221.2.15 rx_etherStatsPkts512to1023Octets

`vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsPkts512to1023Octets`

Rx 512-1023 byte packet

Definition at line 124 of file port.h.

7.221.2.16 rx_etherStatsPkts1024to1518Octets

`vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsPkts1024to1518Octets`

Rx 1024-1518 byte packets

Definition at line 125 of file port.h.

7.221.2.17 rx_etherStatsPkts1519toMaxOctets

`vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsPkts1519toMaxOctets`

Rx 1519- byte packets

Definition at line 126 of file port.h.

7.221.2.18 tx_etherStatsDropEvents

`vtss_port_counter_t vtss_port_rmon_counters_t::tx_etherStatsDropEvents`

Tx drop events

Definition at line 128 of file port.h.

7.221.2.19 tx_etherStatsOctets

`vtss_port_counter_t vtss_port_rmon_counters_t::tx_etherStatsOctets`

Tx octets

Definition at line 129 of file port.h.

7.221.2.20 tx_etherStatsPkts

```
vtss_port_counter_t vtss_port_rmon_counters_t::tx_etherStatsPkts
```

Tx packets

Definition at line 130 of file port.h.

7.221.2.21 tx_etherStatsBroadcastPkts

```
vtss_port_counter_t vtss_port_rmon_counters_t::tx_etherStatsBroadcastPkts
```

Tx broadcasts

Definition at line 131 of file port.h.

7.221.2.22 tx_etherStatsMulticastPkts

```
vtss_port_counter_t vtss_port_rmon_counters_t::tx_etherStatsMulticastPkts
```

Tx multicasts

Definition at line 132 of file port.h.

7.221.2.23 tx_etherStatsCollisions

```
vtss_port_counter_t vtss_port_rmon_counters_t::tx_etherStatsCollisions
```

Tx collisions

Definition at line 133 of file port.h.

7.221.2.24 tx_etherStatsPkts64Octets

```
vtss_port_counter_t vtss_port_rmon_counters_t::tx_etherStatsPkts64Octets
```

Tx 64 byte packets

Definition at line 134 of file port.h.

7.221.2.25 tx_etherStatsPkts65to127Octets

`vtss_port_counter_t vtss_port_rmon_counters_t::tx_etherStatsPkts65to127Octets`

Tx 65-127 byte packets

Definition at line 135 of file port.h.

7.221.2.26 tx_etherStatsPkts128to255Octets

`vtss_port_counter_t vtss_port_rmon_counters_t::tx_etherStatsPkts128to255Octets`

Tx 128-255 byte packets

Definition at line 136 of file port.h.

7.221.2.27 tx_etherStatsPkts256to511Octets

`vtss_port_counter_t vtss_port_rmon_counters_t::tx_etherStatsPkts256to511Octets`

Tx 256-511 byte packets

Definition at line 137 of file port.h.

7.221.2.28 tx_etherStatsPkts512to1023Octets

`vtss_port_counter_t vtss_port_rmon_counters_t::tx_etherStatsPkts512to1023Octets`

Tx 512-1023 byte packet

Definition at line 138 of file port.h.

7.221.2.29 tx_etherStatsPkts1024to1518Octets

`vtss_port_counter_t vtss_port_rmon_counters_t::tx_etherStatsPkts1024to1518Octets`

Tx 1024-1518 byte packets

Definition at line 139 of file port.h.

7.221.2.30 tx_etherStatsPkts1519toMaxOctets

`vtss_port_counter_t vtss_port_rmon_counters_t::tx_etherStatsPkts1519toMaxOctets`

Tx 1519- byte packets

Definition at line 140 of file port.h.

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

- [vtss_api/include/vtss/api/port.h](#)

7.222 vtss_port_serdes_conf_t Struct Reference

SFI Serdes configuration.

```
#include <vtss_port_api.h>
```

Data Fields

- [BOOL sfp_dac](#)

7.222.1 Detailed Description

SFI Serdes configuration.

Definition at line 261 of file vtss_port_api.h.

7.222.2 Field Documentation

7.222.2.1 sfp_dac

`BOOL vtss_port_serdes_conf_t::sfp_dac`

Optical (0) or Cu cable (SFP+ DAC) (1)

Definition at line 263 of file vtss_port_api.h.

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

- [vtss_api/include/vtss_port_api.h](#)

7.223 vtss_port_sgmii_aneg_t Struct Reference

Advertisement control data for SGMII aneg.

```
#include <vtss_port_api.h>
```

Data Fields

- `BOOL link`
- `BOOL fdx`
- `BOOL hdx`
- `BOOL speed_10M`
- `BOOL speed_100M`
- `BOOL speed_1G`
- `BOOL aneg_complete`

7.223.1 Detailed Description

Advertisement control data for SGMII aneg.

Definition at line 139 of file vtss_port_api.h.

7.223.2 Field Documentation

7.223.2.1 link

```
BOOL vtss_port_sgmii_aneg_t::link
```

LP link status

Definition at line 141 of file vtss_port_api.h.

7.223.2.2 fdx

```
BOOL vtss_port_sgmii_aneg_t::fdx
```

FD

Definition at line 142 of file vtss_port_api.h.

7.223.2.3 hdx

`BOOL vtss_port_sgmii_aneg_t::hdx`

HD

Definition at line 143 of file `vtss_port_api.h`.

7.223.2.4 speed_10M

`BOOL vtss_port_sgmii_aneg_t::speed_10M`

speed 10 advertised

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

7.223.2.5 speed_100M

`BOOL vtss_port_sgmii_aneg_t::speed_100M`

speed 100 advertised

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

7.223.2.6 speed_1G

`BOOL vtss_port_sgmii_aneg_t::speed_1G`

speed 1G advertised

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

7.223.2.7 aneg_complete

`BOOL vtss_port_sgmii_aneg_t::aneg_complete`

Aneg process completed

Definition at line 147 of file `vtss_port_api.h`.

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

- `vtss_api/include/vtss_port_api.h`

7.224 vtss_port_status_t Struct Reference

Port status parameter struct.

```
#include <port.h>
```

Data Fields

- [vtss_event_t link_down](#)
- [BOOL link](#)
- [vtss_port_speed_t speed](#)
- [BOOL fdx](#)
- [BOOL remote_fault](#)
- [BOOL aneg_complete](#)
- [BOOL unidirectional_ability](#)
- [vtss_aneg_t aneg](#)
- [BOOL mdi_cross](#)
- [BOOL fiber](#)
- [BOOL copper](#)

7.224.1 Detailed Description

Port status parameter struct.

Definition at line 295 of file port.h.

7.224.2 Field Documentation

7.224.2.1 link_down

```
vtss_event_t vtss_port_status_t::link_down
```

Link down event occurred since last call

Definition at line 297 of file port.h.

7.224.2.2 link

```
BOOL vtss_port_status_t::link
```

Link is up. Remaining fields only valid if TRUE

Definition at line 298 of file port.h.

7.224.2.3 speed

`vtss_port_speed_t` `vtss_port_status_t::speed`

Speed

Definition at line 299 of file port.h.

7.224.2.4 fdx

`BOOL` `vtss_port_status_t::fdx`

Full duplex

Definition at line 300 of file port.h.

7.224.2.5 remote_fault

`BOOL` `vtss_port_status_t::remote_fault`

Remote fault signalled

Definition at line 301 of file port.h.

7.224.2.6 aneg_complete

`BOOL` `vtss_port_status_t::aneg_complete`

Autoneg completed (for clause_37 and Cisco aneg)

Definition at line 302 of file port.h.

7.224.2.7 unidirectional_ability

`BOOL` `vtss_port_status_t::unidirectional_ability`

TRUE: PHY able to transmit from media independent interface regardless of whether the PHY has determined that a valid link has been established.FALSE: PHY able to transmit from media independent interface only when the PHY has determined that a valid link has been established. Note This bit is only applicable to 100BASE-FX and 1000BASE-X fiber media modes.

Definition at line 303 of file port.h.

7.224.2.8 aneg

`vtss_aneg_t vtss_port_status_t::aneg`

Auto negotiation result

Definition at line 307 of file port.h.

7.224.2.9 mdi_cross

`BOOL vtss_port_status_t::mdi_cross`

Indication of if Auto-MDIX crossover is performed

Definition at line 308 of file port.h.

7.224.2.10 fiber

`BOOL vtss_port_status_t::fiber`

Indication of if the link is a fiber link, TRUE if link is a fiber link. FALSE if link is cu link or No Media

Definition at line 309 of file port.h.

7.224.2.11 copper

`BOOL vtss_port_status_t::copper`

Indication of if the link is a copper link, TRUE if link is a copper link. FALSE if link is fiber link or No Media

Definition at line 310 of file port.h.

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

- `vtss_api/include/vtss/api/port.h`

7.225 vtss_qce_action_t Struct Reference

QCE action.

```
#include <vtss_qos_api.h>
```

Data Fields

- `BOOL prio_enable`
- `vtss_prio_t prio`
- `BOOL dp_enable`
- `vtss_dp_level_t dp`
- `BOOL dscp_enable`
- `vtss_dscp_t dscp`

7.225.1 Detailed Description

QCE action.

Definition at line 566 of file `vtss_qos_api.h`.

7.225.2 Field Documentation

7.225.2.1 prio_enable

`BOOL vtss_qce_action_t::prio_enable`

Enable priority classification

Definition at line 568 of file `vtss_qos_api.h`.

7.225.2.2 prio

`vtss_prio_t vtss_qce_action_t::prio`

Priority value

Definition at line 569 of file `vtss_qos_api.h`.

7.225.2.3 dp_enable

`BOOL vtss_qce_action_t::dp_enable`

Enable DP classification

Definition at line 570 of file `vtss_qos_api.h`.

7.225.2.4 dp

`vtss_dp_level_t` `vtss_qce_action_t::dp`

DP value

Definition at line 571 of file `vtss_qos_api.h`.

7.225.2.5 dscp_enable

`BOOL` `vtss_qce_action_t::dscp_enable`

Enable DSCP classification

Definition at line 572 of file `vtss_qos_api.h`.

7.225.2.6 dscp

`vtss_dscp_t` `vtss_qce_action_t::dscp`

DSCP value

Definition at line 573 of file `vtss_qos_api.h`.

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

- `vtss_api/include/vtss_qos_api.h`

7.226 vtss_qce_frame_etype_t Struct Reference

Frame data for VTSS_QCE_TYPE_ETYPE.

```
#include <vtss_qos_api.h>
```

Data Fields

- `vtss_vcap_u16_t` `etype`
- `vtss_vcap_u32_t` `data`

7.226.1 Detailed Description

Frame data for VTSS_QCE_TYPE_ETYPE.

Definition at line 494 of file `vtss_qos_api.h`.

7.226.2 Field Documentation

7.226.2.1 etype

`vtss_vcap_u16_t vtss_qce_frame_etype_t::etype`

Ethernet Type value

Definition at line 496 of file `vtss_qos_api.h`.

7.226.2.2 data

`vtss_vcap_u32_t vtss_qce_frame_etype_t::data`

MAC data

Definition at line 497 of file `vtss_qos_api.h`.

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

- `vtss_api/include/vtss_qos_api.h`

7.227 vtss_qce_frame_ipv4_t Struct Reference

Frame data for VTSS_QCE_TYPE_IPV4.

```
#include <vtss_qos_api.h>
```

Data Fields

- `vtss_vcap_bit_t fragment`
- `vtss_vcap_vr_t dscp`
- `vtss_vcap_u8_t proto`
- `vtss_vcap_ip_t sip`
- `vtss_vcap_vr_t sport`
- `vtss_vcap_vr_t dport`

7.227.1 Detailed Description

Frame data for VTSS_QCE_TYPE_IPV4.

Definition at line 513 of file `vtss_qos_api.h`.

7.227.2 Field Documentation

7.227.2.1 fragment

`vtss_vcap_bit_t` vtss_qce_frame_ipv4_t::fragment

Fragment

Definition at line 515 of file vtss_qos_api.h.

7.227.2.2 dscp

`vtss_vcap_vr_t` vtss_qce_frame_ipv4_t::dscp

DSCP field (6 bit)

Definition at line 516 of file vtss_qos_api.h.

7.227.2.3 proto

`vtss_vcap_u8_t` vtss_qce_frame_ipv4_t::proto

Protocol

Definition at line 517 of file vtss_qos_api.h.

7.227.2.4 sip

`vtss_vcap_ip_t` vtss_qce_frame_ipv4_t::sip

Source IP address - Serval: key_type = normal, ip_addr and mac_ip_addr

Definition at line 518 of file vtss_qos_api.h.

7.227.2.5 sport

`vtss_vcap_vr_t` vtss_qce_frame_ipv4_t::sport

UDP/TCP: Source port - Serval: key_type = normal, ip_addr and mac_ip_addr

Definition at line 522 of file vtss_qos_api.h.

7.227.2.6 dport

`vtss_vcap_vr_t vtss_qce_frame_ipv4_t::dport`

UDP/TCP: Destination port - Serval: key_type = double_tag, ip_addr and mac_ip_addr

Definition at line 523 of file vtss_qos_api.h.

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

- `vtss_api/include/vtss_qos_api.h`

7.228 vtss_qce_frame_ipv6_t Struct Reference

Frame data for VTSS_QCE_TYPE_IPV6.

```
#include <vtss_qos_api.h>
```

Data Fields

- `vtss_vcap_vr_t dscp`
- `vtss_vcap_u8_t proto`
- `vtss_vcap_u128_t sip`
- `vtss_vcap_vr_t sport`
- `vtss_vcap_vr_t dport`

7.228.1 Detailed Description

Frame data for VTSS_QCE_TYPE_IPV6.

Definition at line 527 of file vtss_qos_api.h.

7.228.2 Field Documentation

7.228.2.1 dscp

`vtss_vcap_vr_t vtss_qce_frame_ipv6_t::dscp`

DSCP field (6 bit)

Definition at line 529 of file vtss_qos_api.h.

7.228.2.2 proto

`vtss_vcap_u8_t` `vtss_qce_frame_ipv6_t::proto`

Protocol

Definition at line 530 of file `vtss_qos_api.h`.

7.228.2.3 sip

`vtss_vcap_u128_t` `vtss_qce_frame_ipv6_t::sip`

Source IP address (32 LSB on L26 and J1, 64 LSB on Serval when `key_type = mac_ip_addr`)

Definition at line 531 of file `vtss_qos_api.h`.

7.228.2.4 sport

`vtss_vcap_vr_t` `vtss_qce_frame_ipv6_t::sport`

UDP/TCP: Source port - Serval: `key_type = normal, ip_addr` and `mac_ip_addr`

Definition at line 535 of file `vtss_qos_api.h`.

7.228.2.5 dport

`vtss_vcap_vr_t` `vtss_qce_frame_ipv6_t::dport`

UDP/TCP: Destination port - Serval: `key_type = double_tag, ip_addr` and `mac_ip_addr`

Definition at line 536 of file `vtss_qos_api.h`.

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

- `vtss_api/include/vtss_qos_api.h`

7.229 vtss_qce_frame_llc_t Struct Reference

Frame data for VTSS_QCE_TYPE LLC.

```
#include <vtss_qos_api.h>
```

Data Fields

- `vtss_vcap_u48_t` data

7.229.1 Detailed Description

Frame data for VTSS_QCE_TYPE_LLC.

Definition at line 501 of file `vtss_qos_api.h`.

7.229.2 Field Documentation

7.229.2.1 data

`vtss_vcap_u48_t` `vtss_qce_frame_llc_t::data`

Data

Definition at line 503 of file `vtss_qos_api.h`.

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

- `vtss_api/include/vtss_qos_api.h`

7.230 `vtss_qce_frame_snap_t` Struct Reference

Frame data for VTSS_QCE_TYPE_SNAP.

```
#include <vtss_qos_api.h>
```

Data Fields

- `vtss_vcap_u48_t` data

7.230.1 Detailed Description

Frame data for VTSS_QCE_TYPE_SNAP.

Definition at line 507 of file `vtss_qos_api.h`.

7.230.2 Field Documentation

7.230.2.1 data

```
vtss_vcap_u48_t vtss_qce_frame_snap_t::data
```

Data

Definition at line 509 of file vtss_qos_api.h.

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

- vtss_api/include/vtss_qos_api.h

7.231 vtss_qce_key_t Struct Reference

QCE key.

```
#include <vtss_qos_api.h>
```

Data Fields

- `BOOL port_list [VTSS_PORT_ARRAY_SIZE]`
- `vtss_qce_mac_t mac`
- `vtss_qce_tag_t tag`
- `vtss_qce_type_t type`
- union {
 - `vtss_qce_frame_etype_t etype`
 - `vtss_qce_frame_llc_t llc`
 - `vtss_qce_frame_snap_t snap`
 - `vtss_qce_frame_ipv4_t ipv4`
 - `vtss_qce_frame_ipv6_t ipv6`}
- `frame`

7.231.1 Detailed Description

QCE key.

Definition at line 542 of file vtss_qos_api.h.

7.231.2 Field Documentation

7.231.2.1 port_list

```
BOOL vtss_qce_key_t::port_list [VTSS_PORT_ARRAY_SIZE]
```

Port list

Definition at line 544 of file vtss_qos_api.h.

7.231.2.2 mac

`vtss_qce_mac_t` `vtss_qce_key_t::mac`

MAC

Definition at line 545 of file vtss_qos_api.h.

7.231.2.3 tag

`vtss_qce_tag_t` `vtss_qce_key_t::tag`

Tag

Definition at line 546 of file vtss_qos_api.h.

7.231.2.4 type

`vtss_qce_type_t` `vtss_qce_key_t::type`

Frame type

Definition at line 550 of file vtss_qos_api.h.

7.231.2.5 etype

`vtss_qce_frame_etype_t` `vtss_qce_key_t::etype`

VTSS_QCE_TYPE_ETYPE

Definition at line 555 of file vtss_qos_api.h.

7.231.2.6 llc

`vtss_qce_frame_llc_t` `vtss_qce_key_t::llc`

VTSS_QCE_TYPE_LLCC

Definition at line 556 of file vtss_qos_api.h.

7.231.2.7 snap

`vtss_qce_frame_snap_t vtss_qce_key_t::snap`

VTSS_QCE_TYPE_SNAP

Definition at line 557 of file `vtss_qos_api.h`.

7.231.2.8 ipv4

`vtss_qce_frame_ipv4_t vtss_qce_key_t::ipv4`

VTSS_QCE_TYPE_IPV4

Definition at line 558 of file `vtss_qos_api.h`.

7.231.2.9 ipv6

`vtss_qce_frame_ipv6_t vtss_qce_key_t::ipv6`

VTSS_QCE_TYPE_IPV6

Definition at line 559 of file `vtss_qos_api.h`.

7.231.2.10 frame

`union { ... } vtss_qce_key_t::frame`

Frame type specific data

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

- `vtss_api/include/vtss_qos_api.h`

7.232 vtss_qce_mac_t Struct Reference

QCE MAC information.

```
#include <vtss_qos_api.h>
```

Data Fields

- `vtss_vcap_bit_t dmac_mc`
- `vtss_vcap_bit_t dmac_bc`
- `vtss_vcap_u48_t smac`

7.232.1 Detailed Description

QCE MAC information.

Definition at line 471 of file `vtss_qos_api.h`.

7.232.2 Field Documentation

7.232.2.1 dmac_mc

`vtss_vcap_bit_t vtss_qce_mac_t::dmac_mc`

Multicast DMAC

Definition at line 473 of file `vtss_qos_api.h`.

7.232.2.2 dmac_bc

`vtss_vcap_bit_t vtss_qce_mac_t::dmac_bc`

Broadcast DMAC

Definition at line 474 of file `vtss_qos_api.h`.

7.232.2.3 smac

`vtss_vcap_u48_t vtss_qce_mac_t::smac`

SMAC - Only the 24 most significant bits (OUI) are supported on Jaguar1, rest are wildcards

Definition at line 478 of file `vtss_qos_api.h`.

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

- `vtss_api/include/vtss_qos_api.h`

7.233 vtss_qce_t Struct Reference

QoS Control Entry.

```
#include <vtss_qos_api.h>
```

Data Fields

- `vtss_qce_id_t id`
- `vtss_qce_key_t key`
- `vtss_qce_action_t action`

7.233.1 Detailed Description

QoS Control Entry.

Definition at line 588 of file vtss_qos_api.h.

7.233.2 Field Documentation

7.233.2.1 id

```
vtss_qce_id_t vtss_qce_t::id
```

Entry ID

Definition at line 590 of file vtss_qos_api.h.

7.233.2.2 key

```
vtss_qce_key_t vtss_qce_t::key
```

QCE key

Definition at line 591 of file vtss_qos_api.h.

7.233.2.3 action

`vtss_qce_action_t` `vtss_qce_t::action`

QCE action

Definition at line 592 of file vtss_qos_api.h.

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

- `vtss_api/include/vtss_qos_api.h`

7.234 `vtss_qce_tag_t` Struct Reference

QCE tag information.

```
#include <vtss_qos_api.h>
```

Data Fields

- `vtss_vcap_vr_t vid`
- `vtss_vcap_u8_t pcp`
- `vtss_vcap_bit_t dei`
- `vtss_vcap_bit_t tagged`

7.234.1 Detailed Description

QCE tag information.

Definition at line 482 of file vtss_qos_api.h.

7.234.2 Field Documentation

7.234.2.1 vid

`vtss_vcap_vr_t` `vtss_qce_tag_t::vid`

VLAN ID (12 bit)

Definition at line 484 of file vtss_qos_api.h.

7.234.2.2 pcp

`vtss_vcap_u8_t` `vtss_qce_tag_t::pcp`

PCP (3 bit)

Definition at line 485 of file vtss_qos_api.h.

7.234.2.3 dei

`vtss_vcap_bit_t` `vtss_qce_tag_t::dei`

DEI

Definition at line 486 of file vtss_qos_api.h.

7.234.2.4 tagged

`vtss_vcap_bit_t` `vtss_qce_tag_t::tagged`

Tagged/untagged frame

Definition at line 487 of file vtss_qos_api.h.

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

- vtss_api/include/vtss_qos_api.h

7.235 vtss_qos_conf_t Struct Reference

All parameters below are defined per chip.

```
#include <vtss_qos_api.h>
```

Data Fields

- `vtss_prio_t` prios
- `BOOL` dscp_trust [64]
- `vtss_prio_t` dscp_qos_class_map [64]
- `vtss_dp_level_t` dscp_dp_level_map [64]
- `vtss_dscp_t` dscp_qos_map [VTSS_PRIO_ARRAY_SIZE]
- `BOOL` dscp_remark [64]
- `vtss_dscp_t` dscp_translate_map [64]
- `vtss_dscp_t` dscp_remap [64]

7.235.1 Detailed Description

All parameters below are defined per chip.

Definition at line 91 of file vtss_qos_api.h.

7.235.2 Field Documentation

7.235.2.1 prios

`vtss_prio_t vtss_qos_conf_t::prios`

Number of priorities (1/2/4/8)

Definition at line 93 of file vtss_qos_api.h.

7.235.2.2 dscp_trust

`BOOL vtss_qos_conf_t::dscp_trust[64]`

Ingress: Only trusted DSCP values are used for QOS class and DP level classification

Definition at line 96 of file vtss_qos_api.h.

7.235.2.3 dscp_qos_class_map

`vtss_prio_t vtss_qos_conf_t::dscp_qos_class_map[64]`

Ingress: Mapping from DSCP value to QOS class

Definition at line 97 of file vtss_qos_api.h.

7.235.2.4 dscp_dp_level_map

`vtss_dp_level_t vtss_qos_conf_t::dscp_dp_level_map[64]`

Ingress: Mapping from DSCP value to DP level

Definition at line 98 of file vtss_qos_api.h.

7.235.2.5 dscp_qos_map

```
vtss_dscp_t vtss_qos_conf_t::dscp_qos_map[VTSS_PRIO_ARRAY_SIZE]
```

Ingress: Mapping from QoS class to DSCP (DP unaware or DP level = 0)

Definition at line 100 of file vtss_qos_api.h.

7.235.2.6 dscp_remark

```
BOOL vtss_qos_conf_t::dscp_remark[64]
```

Ingress: DSCP remarking enable. Used when port.dscp_mode = VTSS_DSCP_MODE_SEL

Definition at line 111 of file vtss_qos_api.h.

7.235.2.7 dscp_translate_map

```
vtss_dscp_t vtss_qos_conf_t::dscp_translate_map[64]
```

Ingress: Translated DSCP value. Used when port.dscp_translate = TRUE)

Definition at line 113 of file vtss_qos_api.h.

7.235.2.8 dscp_remap

```
vtss_dscp_t vtss_qos_conf_t::dscp_remap[64]
```

Egress: Remap one DSCP to another (DP unaware or DP level = 0)

Definition at line 114 of file vtss_qos_api.h.

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

- vtss_api/include/[vtss_qos_api.h](#)

7.236 vtss_qos_port_conf_t Struct Reference

QoS setup per port.

```
#include <vtss_qos_api.h>
```

Data Fields

- `vtss_red_t red [VTSS_QUEUE_ARRAY_SIZE]`
- `vtss_policer_t policer_port [VTSS_PORT_POLICERS]`
- `vtss_policer_ext_t policer_ext_port [VTSS_PORT_POLICERS]`
- `vtss_policer_t policer_queue [VTSS_QUEUE_ARRAY_SIZE]`
- `vtss_shaper_t shaper_port`
- `vtss_shaper_t shaper_queue [VTSS_QUEUE_ARRAY_SIZE]`
- `BOOL excess_enable [VTSS_QUEUE_ARRAY_SIZE]`
- `vtss_prio_t default_prio`
- `vtss_tagprio_t usr_prio`
- `vtss_dp_level_t default_dpl`
- `vtss_dei_t default_dei`
- `BOOL tag_class_enable`
- `vtss_prio_t qos_class_map [VTSS_PCP_ARRAY_SIZE][VTSS_DEI_ARRAY_SIZE]`
- `vtss_dp_level_t dp_level_map [VTSS_PCP_ARRAY_SIZE][VTSS_DEI_ARRAY_SIZE]`
- `BOOL dscp_class_enable`
- `vtss_dscp_mode_t dscp_mode`
- `vtss_dscp_emode_t dscp_emode`
- `BOOL dscp_translate`
- `vtss_tag_remark_mode_t tag_remark_mode`
- `vtss_tagprio_t tag_default_pcp`
- `vtss_dei_t tag_default_dei`
- `vtss_tagprio_t tag_pcp_map [VTSS_PRIO_ARRAY_SIZE][2]`
- `vtss_dei_t tag_dei_map [VTSS_PRIO_ARRAY_SIZE][2]`
- `BOOL dwrr_enable`
- `vtss_pct_t queue_pct [VTSS_QUEUE_ARRAY_SIZE]`

7.236.1 Detailed Description

QoS setup per port.

Definition at line 344 of file `vtss_qos_api.h`.

7.236.2 Field Documentation

7.236.2.1 red

```
vtss_red_t vtss_qos_port_conf_t::red[VTSS_QUEUE_ARRAY_SIZE]
```

Random Early Detection

Definition at line 347 of file `vtss_qos_api.h`.

7.236.2.2 policer_port

```
vtss_policer_t vtss_qos_port_conf_t::policer_port[VTSS_PORT_POLICERS]
```

Ingress port policers

Definition at line 350 of file vtss_qos_api.h.

7.236.2.3 policer_ext_port

```
vtss_policer_ext_t vtss_qos_port_conf_t::policer_ext_port[VTSS_PORT_POLICERS]
```

Ingress port policers extensions

Definition at line 353 of file vtss_qos_api.h.

7.236.2.4 policer_queue

```
vtss_policer_t vtss_qos_port_conf_t::policer_queue[VTSS_QUEUE_ARRAY_SIZE]
```

Ingress queue policers

Definition at line 357 of file vtss_qos_api.h.

7.236.2.5 shaper_port

```
vtss_shaper_t vtss_qos_port_conf_t::shaper_port
```

Egress port shaper

Definition at line 360 of file vtss_qos_api.h.

7.236.2.6 shaper_queue

```
vtss_shaper_t vtss_qos_port_conf_t::shaper_queue[VTSS_QUEUE_ARRAY_SIZE]
```

Egress queue shapers

Definition at line 363 of file vtss_qos_api.h.

7.236.2.7 excess_enable

`BOOL vtss_qos_port_conf_t::excess_enable[VTSS_QUEUE_ARRAY_SIZE]`

Allow this queue to use excess bandwidth

Definition at line 365 of file vtss_qos_api.h.

7.236.2.8 default_prio

`vtss_prio_t vtss_qos_port_conf_t::default_prio`

Default port priority (QoS class)

Definition at line 370 of file vtss_qos_api.h.

7.236.2.9 usr_prio

`vtss_tagprio_t vtss_qos_port_conf_t::usr_prio`

Default Ingress VLAN tag priority (PCP)

Definition at line 371 of file vtss_qos_api.h.

7.236.2.10 default_dpl

`vtss_dp_level_t vtss_qos_port_conf_t::default_dpl`

Default Ingress Drop Precedence level

Definition at line 375 of file vtss_qos_api.h.

7.236.2.11 default_dei

`vtss_dei_t vtss_qos_port_conf_t::default_dei`

Default Ingress DEI value

Definition at line 376 of file vtss_qos_api.h.

7.236.2.12 tag_class_enable

`BOOL vtss_qos_port_conf_t::tag_class_enable`

Ingress classification of QoS class and DP level based PCP and DEI

Definition at line 377 of file vtss_qos_api.h.

7.236.2.13 qos_class_map

`vtss_prio_t vtss_qos_port_conf_t::qos_class_map[VTSS_PCP_ARRAY_SIZE] [VTSS_DEI_ARRAY_SIZE]`

Ingress mapping for tagged frames from PCP and DEI to QOS class

Definition at line 378 of file vtss_qos_api.h.

7.236.2.14 dp_level_map

`vtss_dp_level_t vtss_qos_port_conf_t::dp_level_map[VTSS_PCP_ARRAY_SIZE] [VTSS_DEI_ARRAY_SIZE]`

Ingress mapping for tagged frames from PCP and DEI to DP level

Definition at line 379 of file vtss_qos_api.h.

7.236.2.15 dscp_class_enable

`BOOL vtss_qos_port_conf_t::dscp_class_enable`

Ingress classification of QoS class and DP level based on DSCP

Definition at line 380 of file vtss_qos_api.h.

7.236.2.16 dscp_mode

`vtss_dscp_mode_t vtss_qos_port_conf_t::dscp_mode`

Ingress DSCP mode

Definition at line 384 of file vtss_qos_api.h.

7.236.2.17 dscp_emode

`vtss_dscp_emode_t` `vtss_qos_port_conf_t::dscp_emode`

Egress DSCP mode

Definition at line 386 of file vtss_qos_api.h.

7.236.2.18 dscp_translate

`BOOL` `vtss_qos_port_conf_t::dscp_translate`

Ingress: Translate DSCP value via `dscp_translate_map[DSCP]` before use

Definition at line 387 of file vtss_qos_api.h.

7.236.2.19 tag_remark_mode

`vtss_tag_remark_mode_t` `vtss_qos_port_conf_t::tag_remark_mode`

Egress tag remark mode

Definition at line 392 of file vtss_qos_api.h.

7.236.2.20 tag_default_pcp

`vtss_tagprio_t` `vtss_qos_port_conf_t::tag_default_pcp`

Default PCP value for Egress port

Definition at line 393 of file vtss_qos_api.h.

7.236.2.21 tag_default_dei

`vtss_dei_t` `vtss_qos_port_conf_t::tag_default_dei`

Default DEI value for Egress port

Definition at line 394 of file vtss_qos_api.h.

7.236.2.22 tag_pcp_map

```
vtss_tagprior_t vtss_qos_port_conf_t::tag_pcp_map[VTSS_PRIO_ARRAY_SIZE] [2]
```

Egress mapping from QOS class and (1 bit) DP level to PCP

Definition at line 395 of file vtss_qos_api.h.

7.236.2.23 tag_dei_map

```
vtss_dei_t vtss_qos_port_conf_t::tag_dei_map[VTSS_PRIO_ARRAY_SIZE] [2]
```

Egress mapping from QOS class and (1 bit) DP level to DEI

Definition at line 396 of file vtss_qos_api.h.

7.236.2.24 dwrr_enable

```
BOOL vtss_qos_port_conf_t::dwrr_enable
```

Enable Weighted fairness queueing

Definition at line 400 of file vtss_qos_api.h.

7.236.2.25 queue_pct

```
vtss_pct_t vtss_qos_port_conf_t::queue_pct[VTSS_QUEUE_ARRAY_SIZE]
```

Queue percentages

Definition at line 404 of file vtss_qos_api.h.

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

- vtss_api/include/vtss_qos_api.h

7.237 vtss_qs_conf_t Struct Reference

Queue System settings.

```
#include <vtss_init_api.h>
```

Data Fields

- `vtss_qs_mode_t mode`
- `vtss_pct_t oversubscription`
- `struct {`
 - `u32 port_min`
 - `u32 port_max`
 - `u32 queue_min [VTSS_PRIOS]`
 - `u32 queue_max [VTSS_PRIOS]``} port [VTSS_PORTS]`

The settings for each API port and queue in the system.

7.237.1 Detailed Description

Queue System settings.

Definition at line 399 of file vtss_init_api.h.

7.237.2 Field Documentation

7.237.2.1 mode

`vtss_qs_mode_t vtss_qs_conf_t::mode`

The mode of the queue system - default:VTSS_QS_MODE_DISABLED

Definition at line 400 of file vtss_init_api.h.

7.237.2.2 oversubscription

`vtss_pct_t vtss_qs_conf_t::oversubscription`

Queue System oversubscription 0-50%.

Definition at line 401 of file vtss_init_api.h.

7.237.2.3 port_min

`u32 vtss_qs_conf_t::port_min`

Minumum Guaranteed port buffer. Bytes.

Definition at line 405 of file vtss_init_api.h.

7.237.2.4 port_max

```
u32 vtss_qs_conf_t::port_max
```

Maximum port buffer - Not guaranteed. Bytes.

Definition at line 406 of file vtss_init_api.h.

7.237.2.5 queue_min

```
u32 vtss_qs_conf_t::queue_min[VTSS_PRIOS]
```

Minumum Guaranteed queue buffer. Bytes.

Definition at line 407 of file vtss_init_api.h.

7.237.2.6 queue_max

```
u32 vtss_qs_conf_t::queue_max[VTSS_PRIOS]
```

Maximum queue buffer - Not guaranteed. Bytes.

Definition at line 408 of file vtss_init_api.h.

7.237.2.7 port

```
struct { ... } vtss_qs_conf_t::port[VTSS_PORTS]
```

The settings for each API port and queue in the system.

One configuration per port

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

- vtss_api/include/vtss_init_api.h

7.238 vtss_rcpll_status_t Struct Reference

Structure for Get PHY RC-PLL status.

```
#include <vtss_phy_api.h>
```

Data Fields

- `u8 out_of_range`
- `u8 cal_error`
- `u8 cal_not_done`

7.238.1 Detailed Description

Structure for Get PHY RC-PLL status.

Definition at line 1742 of file `vtss_phy_api.h`.

7.238.2 Field Documentation

7.238.2.1 `out_of_range`

```
u8 vtss_rcpll_status_t::out_of_range
```

Out of range condition error

Definition at line 1743 of file `vtss_phy_api.h`.

7.238.2.2 `cal_error`

```
u8 vtss_rcpll_status_t::cal_error
```

Calibration Error indication

Definition at line 1744 of file `vtss_phy_api.h`.

7.238.2.3 `cal_not_done`

```
u8 vtss_rcpll_status_t::cal_not_done
```

Calibration not started or finished

Definition at line 1745 of file `vtss_phy_api.h`.

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

- `vtss_api/include/vtss_phy_api.h`

7.239 `vtss_red_t` Struct Reference

Random Early Detection configuration struct version 1 (per port, per queue)

```
#include <vtss_qos_api.h>
```

Data Fields

- `BOOL enable`
- `vtss_pct_t max_th`
- `vtss_pct_t min_th`
- `vtss_pct_t max_prob_1`
- `vtss_pct_t max_prob_2`
- `vtss_pct_t max_prob_3`

7.239.1 Detailed Description

Random Early Detection configuration struct version 1 (per port, per queue)

Definition at line 48 of file `vtss_qos_api.h`.

7.239.2 Field Documentation

7.239.2.1 `enable`

```
BOOL vtss_red_t::enable
```

Enable/disable RED

Definition at line 50 of file `vtss_qos_api.h`.

7.239.2.2 `max_th`

```
vtss_pct_t vtss_red_t::max_th
```

Maximum threshold

Definition at line 52 of file `vtss_qos_api.h`.

7.239.2.3 min_th

`vtss_pct_t vtss_red_t::min_th`

Minimum threshold

Definition at line 53 of file `vtss_qos_api.h`.

7.239.2.4 max_prob_1

`vtss_pct_t vtss_red_t::max_prob_1`

Drop probability at `max_th` for drop precedence level 1

Definition at line 55 of file `vtss_qos_api.h`.

7.239.2.5 max_prob_2

`vtss_pct_t vtss_red_t::max_prob_2`

Drop probability at `max_th` for drop precedence level 2

Definition at line 56 of file `vtss_qos_api.h`.

7.239.2.6 max_prob_3

`vtss_pct_t vtss_red_t::max_prob_3`

Drop probability at `max_th` for drop precedence level 3

Definition at line 57 of file `vtss_qos_api.h`.

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

- `vtss_api/include/vtss_qos_api.h`

7.240 vtss_restart_status_t Struct Reference

Restart status.

```
#include <vtss_init_api.h>
```

Data Fields

- `vtss_restart_t restart`
- `vtss_version_t prev_version`
- `vtss_version_t cur_version`

7.240.1 Detailed Description

Restart status.

Definition at line 608 of file `vtss_init_api.h`.

7.240.2 Field Documentation

7.240.2.1 restart

```
vtss_restart_t vtss_restart_status_t::restart
```

Previous restart mode

Definition at line 609 of file `vtss_init_api.h`.

7.240.2.2 prev_version

```
vtss_version_t vtss_restart_status_t::prev_version
```

Previous API version

Definition at line 610 of file `vtss_init_api.h`.

7.240.2.3 cur_version

```
vtss_version_t vtss_restart_status_t::cur_version
```

Current API version

Definition at line 611 of file `vtss_init_api.h`.

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

- `vtss_api/include/vtss_init_api.h`

7.241 vtss_routing_entry_t Struct Reference

Routing entry.

```
#include <types.h>
```

Data Fields

- `vtss_routing_entry_type_t type`
- union {
 - `vtss_ipv4_uc_t ipv4_uc`
 - `vtss_ipv6_uc_t ipv6_uc`}
- `vtss_vid_t vlan`

7.241.1 Detailed Description

Routing entry.

Definition at line 868 of file types.h.

7.241.2 Field Documentation

7.241.2.1 type

```
vtss_routing_entry_type_t vtss_routing_entry_t::type
```

Type of route

Definition at line 871 of file types.h.

7.241.2.2 ipv4_uc

```
vtss_ipv4_uc_t vtss_routing_entry_t::ipv4_uc
```

IPv6 unicast route

Definition at line 875 of file types.h.

7.241.2.3 ipv6_uc

`vtss_ipv6_uc_t` `vtss_routing_entry_t::ipv6_uc`

IPv6 unicast route

Definition at line 878 of file types.h.

7.241.2.4 route

`union { ... } vtss_routing_entry_t::route`

Route

7.241.2.5 vlan

`vtss_vid_t` `vtss_routing_entry_t::vlan`

Link-local addresses needs to specify a egress vlan.

Definition at line 882 of file types.h.

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

- `vtss_api/include/vtss/api/types.h`

7.242 vtss_secure_on_passwd_t Struct Reference

Structure for Wake-On-LAN Secure-On Password.

```
#include <vtss_phy_api.h>
```

Data Fields

- `u8 passwd [MAX_WOL_PASSWD_SIZE]`

7.242.1 Detailed Description

Structure for Wake-On-LAN Secure-On Password.

Definition at line 1664 of file vtss_phy_api.h.

7.242.2 Field Documentation

7.242.2.1 passwd

`u8 vtss_secure_on_passwd_t::passwd[MAX_WOL_PASSWD_SIZE]`

Secure-On Password, Can be 4 or 6 bytes

Definition at line 1666 of file vtss_phy_api.h.

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

- `vtss_api/include/vtss_phy_api.h`

7.243 vtss_serdes_macro_conf_t Struct Reference

Serdes macro configuration.

`#include <vtss_init_api.h>`

Data Fields

- `vtss_vdd_t serdes1g_vdd`
- `vtss_vdd_t serdes6g_vdd`
- `BOOL ib_cterm_ena`
- `serdes_fields_t qsgmii`

7.243.1 Detailed Description

Serdes macro configuration.

Definition at line 363 of file vtss_init_api.h.

7.243.2 Field Documentation

7.243.2.1 serdes1g_vdd

`vtss_vdd_t vtss_serdes_macro_conf_t::serdes1g_vdd`

Serdes1g supply

Definition at line 364 of file vtss_init_api.h.

7.243.2.2 serdes6g_vdd

`vtss_vdd_t` `vtss_serdes_macro_conf_t::serdes6g_vdd`

Serdes6g supply

Definition at line 365 of file `vtss_init_api.h`.

7.243.2.3 ib_cterm_ena

`BOOL` `vtss_serdes_macro_conf_t::ib_cterm_ena`

AC(0)/DC(1) coupled

Definition at line 366 of file `vtss_init_api.h`.

7.243.2.4 qsgmii

`serdes_fields_t` `vtss_serdes_macro_conf_t::qsgmii`

Appl/Board specific fields for QSGMII

Definition at line 367 of file `vtss_init_api.h`.

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

- `vtss_api/include/vtss_init_api.h`

7.244 vtss_sflow_port_conf_t Struct Reference

sFlow configuration structure.

```
#include <vtss_l2_api.h>
```

Data Fields

- `vtss_sflow_type_t type`
- `u32 sampling_rate`

7.244.1 Detailed Description

sFlow configuration structure.

Not all sampling rates are supported. Since the sFlow standard mandates that returned sample rates are actual sample rates and not desired sample rates, it is recommended to call [vtss_sflow_sampling_rate_convert\(\)](#) to obtain the actual sample rate given a desired sample rate. [vtss_sflow_port_conf_set\(\)](#) will auto-convert the requested sample rate to an actual sample rate, which will be returned in subsequent calls to [vtss_sflow_port_conf_get\(\)](#).

Definition at line 1450 of file vtss_l2_api.h.

7.244.2 Field Documentation

7.244.2.1 type

```
vtss_sflow_type_t vtss_sflow_port_conf_t::type
```

Sample direction. Also used to turn off sampling.

Definition at line 1451 of file vtss_l2_api.h.

7.244.2.2 sampling_rate

```
u32 vtss_sflow_port_conf_t::sampling_rate
```

A value of N means: sample on average 1 out of N frames. 0 disables sampling.

Definition at line 1452 of file vtss_l2_api.h.

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

- vtss_api/include/[vtss_l2_api.h](#)

7.245 vtss_sgpi_conf_t Struct Reference

SGPIO configuration for a group.

```
#include <vtss_misc_api.h>
```

Data Fields

- [vtss_sgpi_bmode_t bmode](#) [2]
- [u8 bit_count](#)
- [vtss_sgpi_port_conf_t port_conf](#) [[VTSS_SGPIO_PORTS](#)]

7.245.1 Detailed Description

SGPIO configuration for a group.

Definition at line 797 of file vtss_misc_api.h.

7.245.2 Field Documentation

7.245.2.1 bmode

```
vtss_sgpi_bmode_t vtss_sgpi_conf_t::bmode[2]
```

Blink mode 0 and 1

Definition at line 799 of file vtss_misc_api.h.

7.245.2.2 bit_count

```
u8 vtss_sgpi_conf_t::bit_count
```

Bits enabled per port, 1-4

Definition at line 800 of file vtss_misc_api.h.

7.245.2.3 port_conf

```
vtss_sgpi_port_conf_t vtss_sgpi_conf_t::port_conf[VTSS_SGPIO_PORTS]
```

Port configuration

Definition at line 801 of file vtss_misc_api.h.

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

- vtss_api/include/vtss_misc_api.h

7.246 vtss_sgpi_port_conf_t Struct Reference

SGPIO port configuration.

```
#include <vtss_misc_api.h>
```

Data Fields

- `BOOL enabled`
- `vtss_sgpi_mode_t mode [4]`
- `BOOL int_pol_high [4]`

7.246.1 Detailed Description

SGPIO port configuration.

Definition at line 789 of file `vtss_misc_api.h`.

7.246.2 Field Documentation

7.246.2.1 enabled

`BOOL vtss_sgpi_port_conf_t::enabled`

Port enabled/disabled

Definition at line 791 of file `vtss_misc_api.h`.

7.246.2.2 mode

`vtss_sgpi_mode_t vtss_sgpi_port_conf_t::mode [4]`

Mode for each bit

Definition at line 792 of file `vtss_misc_api.h`.

7.246.2.3 int_pol_high

`BOOL vtss_sgpi_port_conf_t::int_pol_high [4]`

SGPIO interrupt polarity for each bit - TRUE - Interrupt when SGPIO pin high, FALSE - Interrupt when SGPIO pin low.

Definition at line 793 of file `vtss_misc_api.h`.

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

- `vtss_api/include/vtss_misc_api.h`

7.247 vtss_sgpi_port_data_t Struct Reference

SGPIO read data for a port.

```
#include <vtss_misc_api.h>
```

Data Fields

- [BOOL value \[4\]](#)

7.247.1 Detailed Description

SGPIO read data for a port.

Definition at line 835 of file vtss_misc_api.h.

7.247.2 Field Documentation

7.247.2.1 value

```
BOOL vtss_sgpi_port_data_t::value[4]
```

Data for each and bit

Definition at line 837 of file vtss_misc_api.h.

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

- vtss_api/include/[vtss_misc_api.h](#)

7.248 vtss_shaper_t Struct Reference

Shaper.

```
#include <vtss_qos_api.h>
```

Data Fields

- [vtss_burst_level_t level](#)
- [vtss_bitrate_t rate](#)

7.248.1 Detailed Description

Shaper.

Definition at line 298 of file vtss_qos_api.h.

7.248.2 Field Documentation

7.248.2.1 level

`vtss_burst_level_t` `vtss_shaper_t::level`

CBS (Committed Burst Size). Unit: bytes

Definition at line 300 of file vtss_qos_api.h.

7.248.2.2 rate

`vtss_bitrate_t` `vtss_shaper_t::rate`

CIR (Committed Information Rate). Unit: kbps. Use VTSS_BITRATE_DISABLED to disable shaper

Definition at line 301 of file vtss_qos_api.h.

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

- vtss_api/include/[vtss_qos_api.h](#)

7.249 vtss_sublayer_status_t Struct Reference

10G Phy link and fault status

```
#include <vtss_phy_10g_api.h>
```

Data Fields

- `BOOL rx_link`
- `vtss_event_t link_down`
- `BOOL rx_fault`
- `BOOL tx_fault`

7.249.1 Detailed Description

10G Phy link and fault status

Definition at line 86 of file `vtss_phy_10g_api.h`.

7.249.2 Field Documentation

7.249.2.1 rx_link

`BOOL vtss_sublayer_status_t::rx_link`

The rx link status

Definition at line 87 of file `vtss_phy_10g_api.h`.

7.249.2.2 link_down

`vtss_event_t vtss_sublayer_status_t::link_down`

Link down event status. Clear on read

Definition at line 88 of file `vtss_phy_10g_api.h`.

7.249.2.3 rx_fault

`BOOL vtss_sublayer_status_t::rx_fault`

Rx fault event status. Clear on read

Definition at line 89 of file `vtss_phy_10g_api.h`.

7.249.2.4 tx_fault

`BOOL vtss_sublayer_status_t::tx_fault`

Tx fault event status. Clear on read

Definition at line 90 of file `vtss_phy_10g_api.h`.

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

- `vtss_api/include/vtss_phy_10g_api.h`

7.250 vtss_sync_clock_in_t Struct Reference

Struct containing configuration selecting the recovered input clock port, to be delivered to a selected output clock port.

```
#include <vtss_sync_api.h>
```

Data Fields

- `vtss_port_no_t port_no`
- `BOOL squelsh`
- `BOOL enable`

7.250.1 Detailed Description

Struct containing configuration selecting the recovered input clock port, to be delivered to a selected output clock port.

Definition at line 107 of file `vtss_sync_api.h`.

7.250.2 Field Documentation

7.250.2.1 port_no

```
vtss_port_no_t vtss_sync_clock_in_t::port_no
```

Selection of the input port number - must map to a SERDES port

Definition at line 109 of file `vtss_sync_api.h`.

7.250.2.2 squelsh

```
BOOL vtss_sync_clock_in_t::squelsh
```

Enable/disable of automatic squelsh

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

7.250.2.3 enable

`BOOL vtss_sync_clock_in_t::enable`

Enable/disable of delivery of recovered clock to this selected output clock port

Definition at line 111 of file vtss_sync_api.h.

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

- `vtss_api/include/vtss_sync_api.h`

7.251 vtss_sync_clock_out_t Struct Reference

Struct containing configuration for a recovered clock output port.

```
#include <vtss_sync_api.h>
```

Data Fields

- `vtss_sync_divider_t divider`
- `BOOL enable`

7.251.1 Detailed Description

Struct containing configuration for a recovered clock output port.

Definition at line 73 of file vtss_sync_api.h.

7.251.2 Field Documentation

7.251.2.1 divider

`vtss_sync_divider_t vtss_sync_clock_out_t::divider`

Selection the clock division. This should be set to VTSS_SYNC_DIVIDER_1 if recovered clock is comming from internal PHY

Definition at line 75 of file vtss_sync_api.h.

7.251.2.2 enable

`BOOL vtss_sync_clock_out_t::enable`

Enable/disable of this output clock port

Definition at line 76 of file `vtss_sync_api.h`.

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

- `vtss_api/include/vtss_sync_api.h`

7.252 `vtss_tci_t` Struct Reference

Tag Control Information (according to IEEE 802.1Q)

```
#include <vtss_packet_api.h>
```

Data Fields

- `vtss_vid_t vid`
- `BOOL cfi`
- `vtss_tagprio_t tagprio`

7.252.1 Detailed Description

Tag Control Information (according to IEEE 802.1Q)

Definition at line 206 of file `vtss_packet_api.h`.

7.252.2 Field Documentation

7.252.2.1 vid

`vtss_vid_t vtss_tci_t::vid`

VLAN ID

Definition at line 208 of file `vtss_packet_api.h`.

7.252.2.2 cfi

`BOOL vtss_tci_t::cfi`

Canonical Format Indicator

Definition at line 209 of file vtss_packet_api.h.

7.252.2.3 tagprio

`vtss_tagprio_t vtss_tci_t::tagprio`

Tag priority

Definition at line 210 of file vtss_packet_api.h.

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

- vtss_api/include/[vtss_packet_api.h](#)

7.253 vtss_timeofday_t Struct Reference

Time of day structure.

```
#include <vtss_os_ecos.h>
```

Data Fields

- `u32 sec`
- `time_t sec`

7.253.1 Detailed Description

Time of day structure.

Definition at line 59 of file vtss_os_ecos.h.

7.253.2 Field Documentation

7.253.2.1 sec [1/2]

u32 vtss_timeofday_t::sec

Time of day in seconds

Definition at line 60 of file vtss_os_ecos.h.

7.253.2.2 sec [2/2]

time_t vtss_timeofday_t::sec

Time of day in seconds

Definition at line 109 of file vtss_os_linux.h.

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

- vtss_api/include/[vtss_os_ecos.h](#)
- vtss_api/include/[vtss_os_linux.h](#)

7.254 vtss_timestamp_t Struct Reference

Time stamp in seconds and nanoseconds.

```
#include <types.h>
```

Data Fields

- u16 sec_msb
- u32 seconds
- u32 nanoseconds

7.254.1 Detailed Description

Time stamp in seconds and nanoseconds.

Definition at line 1212 of file types.h.

7.254.2 Field Documentation

7.254.2.1 sec_msb

`u16 vtss_timestamp_t::sec_msb`

Seconds msb

Definition at line 1213 of file types.h.

7.254.2.2 seconds

`u32 vtss_timestamp_t::seconds`

Seconds

Definition at line 1214 of file types.h.

7.254.2.3 nanoseconds

`u32 vtss_timestamp_t::nanoseconds`

nanoseconds

Definition at line 1215 of file types.h.

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

- `vtss_api/include/vtss/api/types.h`

7.255 vtss_trace_conf_t Struct Reference

Trace group configuration.

```
#include <vtss_misc_api.h>
```

Data Fields

- `vtss_trace_level_t level [VTSS_TRACE_LAYER_COUNT]`

7.255.1 Detailed Description

Trace group configuration.

Definition at line 97 of file vtss_misc_api.h.

7.255.2 Field Documentation

7.255.2.1 level

```
vtss_trace_level_t vtss_trace_conf_t::level[VTSS_TRACE_LAYER_COUNT]
```

Trace level per layer

Definition at line 99 of file vtss_misc_api.h.

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

- vtss_api/include/vtss_misc_api.h

7.256 vtss_ts_ext_clock_mode_t Struct Reference

external clock output configuration.

```
#include <vtss_ts_api.h>
```

Data Fields

- [vtss_ts_ext_clock_one_pps_mode_t one_pps_mode](#)
- [BOOL enable](#)
- [u32 freq](#)

7.256.1 Detailed Description

external clock output configuration.

Definition at line 289 of file vtss_ts_api.h.

7.256.2 Field Documentation

7.256.2.1 one_pps_mode

```
vtss_ts_ext_clock_one_pps_mode_t vtss_ts_ext_clock_mode_t::one_pps_mode
```

Select 1pps ext clock mode: input : lock clock to 1pps input output: enable external sync pulse output disable: disable 1 pps

Definition at line 290 of file vtss_ts_api.h.

7.256.2.2 enable

`BOOL vtss_ts_ext_clock_mode_t::enable`

Select internal sync pulse (enable = false) or external sync pulse (enable = true)

Definition at line 295 of file vtss_ts_api.h.

7.256.2.3 freq

`u32 vtss_ts_ext_clock_mode_t::freq`

clock output frequency (hz [1..25.000.000]).

Definition at line 297 of file vtss_ts_api.h.

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

- vtss_api/include/[vtss_ts_api.h](#)

7.257 vtss_ts_id_t Struct Reference

Timestamp identifier.

```
#include <vtss_ts_api.h>
```

Data Fields

- [u32 ts_id](#)

7.257.1 Detailed Description

Timestamp identifier.

Definition at line 527 of file vtss_ts_api.h.

7.257.2 Field Documentation

7.257.2.1 ts_id

```
u32 vtss_ts_id_t::ts_id
```

Timestamp identifier

Definition at line 528 of file vtss_ts_api.h.

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

- [vtss_api/include/vtss_ts_api.h](#)

7.258 vtss_ts_internal_mode_t Struct Reference

Hardware timestamping format mode for internal ports.

```
#include <vtss_ts_api.h>
```

Data Fields

- [vtss_ts_internal_fmt_t int_fmt](#)

7.258.1 Detailed Description

Hardware timestamping format mode for internal ports.

Definition at line 498 of file vtss_ts_api.h.

7.258.2 Field Documentation

7.258.2.1 int_fmt

```
vtss_ts_internal_fmt_t vtss_ts_internal_mode_t::int_fmt
```

Hardware Timestamping format mode for INTERNAL ports

Definition at line 499 of file vtss_ts_api.h.

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

- [vtss_api/include/vtss_ts_api.h](#)

7.259 vtss_ts_operation_mode_t Struct Reference

Timestamp operation.

```
#include <vtss_ts_api.h>
```

Data Fields

- [vtss_ts_mode_t mode](#)

7.259.1 Detailed Description

Timestamp operation.

Definition at line 451 of file vtss_ts_api.h.

7.259.2 Field Documentation

7.259.2.1 mode

```
vtss_ts_mode_t vtss_ts_operation_mode_t::mode
```

Hardware Timestamping mode for a port(EXTERNAL or INTERNAL)

Definition at line 452 of file vtss_ts_api.h.

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

- [vtss_api/include/vtss_ts_api.h](#)

7.260 vtss_ts_timestamp_alloc_t Struct Reference

Timestamp allocation.

```
#include <vtss_ts_api.h>
```

Data Fields

- [u64 port_mask](#)
- [void * context](#)
- [void\(* cb \)\(void *context, u32 port_no, vtss_ts_timestamp_t *ts\)](#)

7.260.1 Detailed Description

Timestamp allocation.

Definition at line 613 of file vtss_ts_api.h.

7.260.2 Field Documentation

7.260.2.1 port_mask

```
u64 vtss_ts_timestamp_alloc_t::port_mask
```

Identify the ports that a timestamp id is allocated to

Definition at line 614 of file vtss_ts_api.h.

7.260.2.2 context

```
void* vtss_ts_timestamp_alloc_t::context
```

Application specific context used as parameter in the call-out

Definition at line 615 of file vtss_ts_api.h.

7.260.2.3 cb

```
void(* vtss_ts_timestamp_alloc_t::cb) (void *context, u32 port_no, vtss_ts_timestamp_t *ts)
```

Application call-out function called when the timestamp is available

Definition at line 616 of file vtss_ts_api.h.

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

- vtss_api/include/[vtss_ts_api.h](#)

7.261 vtss_ts_timestamp_t Struct Reference

Timestamp structure.

```
#include <vtss_ts_api.h>
```

Data Fields

- `u32 ts`
- `u32 id`
- `void * context`
- `BOOL ts_valid`

7.261.1 Detailed Description

Timestamp structure.

Definition at line 532 of file vtss_ts_api.h.

7.261.2 Field Documentation

7.261.2.1 ts

`u32 vtss_ts_timestamp_t::ts`

Timestamp value

Definition at line 533 of file vtss_ts_api.h.

7.261.2.2 id

`u32 vtss_ts_timestamp_t::id`

Timestamp identifier

Definition at line 534 of file vtss_ts_api.h.

7.261.2.3 context

`void* vtss_ts_timestamp_t::context`

Application specific context

Definition at line 535 of file vtss_ts_api.h.

7.261.2.4 ts_valid

`BOOL vtss_ts_timestamp_t::ts_valid`

Timestamp is valid (can be not valid if timestamp is not received)

Definition at line 536 of file `vtss_ts_api.h`.

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

- `vtss_api/include/vtss_ts_api.h`

7.262 vtss_vcap_ip_t Struct Reference

VCAP IPv4 address value and mask.

```
#include <types.h>
```

Data Fields

- `vtss_ip_t value`
- `vtss_ip_t mask`

7.262.1 Detailed Description

VCAP IPv4 address value and mask.

Definition at line 968 of file `types.h`.

7.262.2 Field Documentation

7.262.2.1 value

`vtss_ip_t vtss_vcap_ip_t::value`

Value

Definition at line 970 of file `types.h`.

7.262.2.2 mask

`vtss_ip_t` `vtss_vcap_ip_t::mask`

Mask, cleared bits are wildcards

Definition at line 971 of file types.h.

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

- `vtss_api/include/vtss/api/types.h`

7.263 vtss_vcap_u128_t Struct Reference

VCAP 128 bit value and mask.

```
#include <types.h>
```

Data Fields

- `u8 value [16]`
- `u8 mask [16]`

7.263.1 Detailed Description

VCAP 128 bit value and mask.

Definition at line 954 of file types.h.

7.263.2 Field Documentation

7.263.2.1 value

`u8 vtss_vcap_u128_t::value[16]`

Value

Definition at line 956 of file types.h.

7.263.2.2 mask

`u8 vtss_vcap_u128_t::mask[16]`

Mask, cleared bits are wildcards

Definition at line 957 of file types.h.

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

- [vtss_api/include/vtss/api/types.h](#)

7.264 vtss_vcap_u16_t Struct Reference

VCAP 16 bit value and mask.

```
#include <types.h>
```

Data Fields

- [u8 value \[2\]](#)
- [u8 mask \[2\]](#)

7.264.1 Detailed Description

VCAP 16 bit value and mask.

Definition at line 919 of file types.h.

7.264.2 Field Documentation

7.264.2.1 value

`u8 vtss_vcap_u16_t::value[2]`

Value

Definition at line 921 of file types.h.

7.264.2.2 mask

`u8 vtss_vcap_u16_t::mask[2]`

Mask, cleared bits are wildcards

Definition at line 922 of file types.h.

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

- [vtss_api/include/vtss/api/types.h](#)

7.265 vtss_vcap_u24_t Struct Reference

VCAP 24 bit value and mask.

```
#include <types.h>
```

Data Fields

- [u8 value \[3\]](#)
- [u8 mask \[3\]](#)

7.265.1 Detailed Description

VCAP 24 bit value and mask.

Definition at line 926 of file types.h.

7.265.2 Field Documentation

7.265.2.1 value

`u8 vtss_vcap_u24_t::value[3]`

Value

Definition at line 928 of file types.h.

7.265.2.2 mask

`u8 vtss_vcap_u24_t::mask [3]`

Mask, cleared bits are wildcards

Definition at line 929 of file types.h.

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

- [vtss_api/include/vtss/api/types.h](#)

7.266 vtss_vcap_u32_t Struct Reference

VCAP 32 bit value and mask.

```
#include <types.h>
```

Data Fields

- [u8 value \[4\]](#)
- [u8 mask \[4\]](#)

7.266.1 Detailed Description

VCAP 32 bit value and mask.

Definition at line 933 of file types.h.

7.266.2 Field Documentation

7.266.2.1 value

`u8 vtss_vcap_u32_t::value[4]`

Value

Definition at line 935 of file types.h.

7.266.2.2 mask

`u8 vtss_vcap_u32_t::mask[4]`

Mask, cleared bits are wildcards

Definition at line 936 of file types.h.

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

- [vtss_api/include/vtss/api/types.h](#)

7.267 vtss_vcap_u40_t Struct Reference

VCAP 40 bit value and mask.

```
#include <types.h>
```

Data Fields

- [u8 value \[5\]](#)
- [u8 mask \[5\]](#)

7.267.1 Detailed Description

VCAP 40 bit value and mask.

Definition at line 940 of file types.h.

7.267.2 Field Documentation

7.267.2.1 value

`u8 vtss_vcap_u40_t::value[5]`

Value

Definition at line 942 of file types.h.

7.267.2.2 mask

`u8 vtss_vcap_u40_t::mask [5]`

Mask, cleared bits are wildcards

Definition at line 943 of file types.h.

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

- `vtss_api/include/vtss/api/types.h`

7.268 vtss_vcap_u48_t Struct Reference

VCAP 48 bit value and mask.

```
#include <types.h>
```

Data Fields

- `u8 value [6]`
- `u8 mask [6]`

7.268.1 Detailed Description

VCAP 48 bit value and mask.

Definition at line 947 of file types.h.

7.268.2 Field Documentation

7.268.2.1 value

`u8 vtss_vcap_u48_t::value [6]`

Value

Definition at line 949 of file types.h.

7.268.2.2 mask

`u8 vtss_vcap_u48_t::mask[6]`

Mask, cleared bits are wildcards

Definition at line 950 of file types.h.

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

- [vtss_api/include/vtss/api/types.h](#)

7.269 vtss_vcap_u8_t Struct Reference

VCAP 8 bit value and mask.

```
#include <types.h>
```

Data Fields

- [u8 value](#)
- [u8 mask](#)

7.269.1 Detailed Description

VCAP 8 bit value and mask.

Definition at line 912 of file types.h.

7.269.2 Field Documentation

7.269.2.1 value

`u8 vtss_vcap_u8_t::value`

Value

Definition at line 914 of file types.h.

7.269.2.2 mask

`u8 vtss_vcap_u8_t::mask`

Mask, cleared bits are wildcards

Definition at line 915 of file types.h.

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

- [vtss_api/include/vtss/api/types.h](#)

7.270 vtss_vcap_udp_tcp_t Struct Reference

VCAP UDP/TCP port range.

```
#include <types.h>
```

Data Fields

- [BOOL in_range](#)
- [vtss_udp_tcp_t low](#)
- [vtss_udp_tcp_t high](#)

7.270.1 Detailed Description

VCAP UDP/TCP port range.

Definition at line 975 of file types.h.

7.270.2 Field Documentation

7.270.2.1 in_range

`BOOL vtss_vcap_udp_tcp_t::in_range`

Port in range match

Definition at line 977 of file types.h.

7.270.2.2 low

`vtss_udp_tcp_t vtss_vcap_udp_tcp_t::low`

Port low value

Definition at line 978 of file types.h.

7.270.2.3 high

`vtss_udp_tcp_t vtss_vcap_udp_tcp_t::high`

Port high value

Definition at line 979 of file types.h.

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

- `vtss_api/include/vtss/api/types.h`

7.271 vtss_vcap_vid_t Struct Reference

VCAP VLAN ID value and mask.

```
#include <types.h>
```

Data Fields

- `u16 value`
- `u16 mask`

7.271.1 Detailed Description

VCAP VLAN ID value and mask.

Definition at line 961 of file types.h.

7.271.2 Field Documentation

7.271.2.1 value

`u16 vtss_vcap_vid_t::value`

Value

Definition at line 963 of file types.h.

7.271.2.2 mask

`u16 vtss_vcap_vid_t::mask`

Mask, cleared bits are wildcards

Definition at line 964 of file types.h.

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

- `vtss_api/include/vtss/api/types.h`

7.272 vtss_vcap_vr_t Struct Reference

VCAP universal value or range.

```
#include <types.h>
```

Data Fields

- `vtss_vcap_vr_type_t type`
- union {
 - struct {
 - `vtss_vcap_vr_value_t value`
 - `vtss_vcap_vr_value_t mask`
 - `} v`
 - struct {
 - `vtss_vcap_vr_value_t low`
 - `vtss_vcap_vr_value_t high`
 - `} r`
- `} vr`

7.272.1 Detailed Description

VCAP universal value or range.

Definition at line 994 of file types.h.

7.272.2 Field Documentation

7.272.2.1 type

`vtss_vcap_vr_type_t vtss_vcap_vr_t::type`

Type

Definition at line 996 of file types.h.

7.272.2.2 value

`vtss_vcap_vr_value_t vtss_vcap_vr_t::value`

Value

Definition at line 1001 of file types.h.

7.272.2.3 mask

`vtss_vcap_vr_value_t vtss_vcap_vr_t::mask`

Mask, cleared bits are wildcards

Definition at line 1002 of file types.h.

7.272.2.4 v

`struct { ... } vtss_vcap_vr_t::v`

`type == VTSS_VCAP_VR_TYPE_VALUE_MASK`

7.272.2.5 low

`vtss_vcap_vr_value_t vtss_vcap_vr_t::low`

Low value

Definition at line 1006 of file types.h.

7.272.2.6 high

```
vtss_vcap_vr_value_t vtss_vcap_vr_t::high
```

High value

Definition at line 1007 of file types.h.

7.272.2.7 r

```
struct { ... } vtss_vcap_vr_t::r  
type == VTSS_VCAP_VR_TYPE_RANGE_XXXXXX
```

7.272.2.8 vr

```
union { ... } vtss_vcap_vr_t::vr
```

Value or range

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

- [vtss_api/include/vtss/api/types.h](#)

7.273 vtss_vce_action_t Struct Reference

VCE Action.

```
#include <vtss_l2_api.h>
```

Data Fields

- [vtss_vid_t vid](#)
- [vtss_acl_policy_no_t policy_no](#)

7.273.1 Detailed Description

VCE Action.

Definition at line 1006 of file vtss_l2_api.h.

7.273.2 Field Documentation

7.273.2.1 vid

`vtss_vid_t` `vtss_vce_action_t::vid`

Classified VLAN ID

Definition at line 1008 of file `vtss_l2_api.h`.

7.273.2.2 policy_no

`vtss_acl_policy_no_t` `vtss_vce_action_t::policy_no`

ACL policy number

Definition at line 1009 of file `vtss_l2_api.h`.

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

- `vtss_api/include/vtss_l2_api.h`

7.274 vtss_vce_frame_etype_t Struct Reference

Frame data for VTSS_VCE_TYPE_ETYPE.

```
#include <vtss_l2_api.h>
```

Data Fields

- `vtss_vcap_u16_t` `etype`
- `vtss_vcap_u32_t` `data`

7.274.1 Detailed Description

Frame data for VTSS_VCE_TYPE_ETYPE.

Definition at line 948 of file `vtss_l2_api.h`.

7.274.2 Field Documentation

7.274.2.1 etype

`vtss_vcap_u16_t vtss_vce_frame_etype_t::etype`

Ethernet Type value

Definition at line 950 of file `vtss_l2_api.h`.

7.274.2.2 data

`vtss_vcap_u32_t vtss_vce_frame_etype_t::data`

MAC data

Definition at line 951 of file `vtss_l2_api.h`.

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

- `vtss_api/include/vtss_l2_api.h`

7.275 vtss_vce_frame_ipv4_t Struct Reference

Frame data for VTSS_VCE_TYPE_IPV4.

```
#include <vtss_l2_api.h>
```

Data Fields

- `vtss_vcap_bit_t fragment`
- `vtss_vcap_bit_t options`
- `vtss_vcap_vr_t dscp`
- `vtss_vcap_u8_t proto`
- `vtss_vcap_ip_t sip`
- `vtss_vcap_vr_t dport`

7.275.1 Detailed Description

Frame data for VTSS_VCE_TYPE_IPV4.

Definition at line 967 of file `vtss_l2_api.h`.

7.275.2 Field Documentation

7.275.2.1 fragment

`vtss_vcap_bit_t` vtss_vce_frame_ipv4_t::fragment

Fragment

Definition at line 969 of file vtss_l2_api.h.

7.275.2.2 options

`vtss_vcap_bit_t` vtss_vce_frame_ipv4_t::options

Header options

Definition at line 970 of file vtss_l2_api.h.

7.275.2.3 dscp

`vtss_vcap_vr_t` vtss_vce_frame_ipv4_t::dscp

DSCP field (6 bit)

Definition at line 971 of file vtss_l2_api.h.

7.275.2.4 proto

`vtss_vcap_u8_t` vtss_vce_frame_ipv4_t::proto

Protocol

Definition at line 972 of file vtss_l2_api.h.

7.275.2.5 sip

`vtss_vcap_ip_t` vtss_vce_frame_ipv4_t::sip

Source IP address

Definition at line 973 of file vtss_l2_api.h.

7.275.2.6 dport

`vtss_vcap_vr_t vtss_vce_frame_ipv4_t::dport`

UDP/TCP: Destination port

Definition at line 974 of file vtss_l2_api.h.

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

- `vtss_api/include/vtss_l2_api.h`

7.276 vtss_vce_frame_ipv6_t Struct Reference

Frame data for VTSS_VCE_TYPE_IPV6.

```
#include <vtss_l2_api.h>
```

Data Fields

- `vtss_vcap_vr_t dscp`
- `vtss_vcap_u8_t proto`
- `vtss_vcap_u32_t sip`
- `vtss_vcap_vr_t dport`

7.276.1 Detailed Description

Frame data for VTSS_VCE_TYPE_IPV6.

Definition at line 978 of file vtss_l2_api.h.

7.276.2 Field Documentation

7.276.2.1 dscp

`vtss_vcap_vr_t vtss_vce_frame_ipv6_t::dscp`

DSCP field (6 bit)

Definition at line 980 of file vtss_l2_api.h.

7.276.2.2 proto

`vtss_vcap_u8_t` `vtss_vce_frame_ipv6_t::proto`

Protocol

Definition at line 981 of file vtss_l2_api.h.

7.276.2.3 sip

`vtss_vcap_u32_t` `vtss_vce_frame_ipv6_t::sip`

Source IP address (32 LSB)

Definition at line 982 of file vtss_l2_api.h.

7.276.2.4 dport

`vtss_vcap_vr_t` `vtss_vce_frame_ipv6_t::dport`

UDP/TCP: Destination port

Definition at line 983 of file vtss_l2_api.h.

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

- `vtss_api/include/vtss_l2_api.h`

7.277 vtss_vce_frame_llc_t Struct Reference

Frame data for VTSS_VCE_TYPE LLC.

```
#include <vtss_l2_api.h>
```

Data Fields

- `vtss_vcap_u48_t data`

7.277.1 Detailed Description

Frame data for VTSS_VCE_TYPE LLC.

Definition at line 955 of file vtss_l2_api.h.

7.277.2 Field Documentation

7.277.2.1 data

`vtss_vcap_u48_t vtss_vce_frame_llc_t::data`

Data

Definition at line 957 of file vtss_l2_api.h.

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

- `vtss_api/include/vtss_l2_api.h`

7.278 vtss_vce_frame_snap_t Struct Reference

Frame data for VTSS_VCE_TYPE_SNAP.

```
#include <vtss_l2_api.h>
```

Data Fields

- `vtss_vcap_u48_t data`

7.278.1 Detailed Description

Frame data for VTSS_VCE_TYPE_SNAP.

Definition at line 961 of file vtss_l2_api.h.

7.278.2 Field Documentation

7.278.2.1 data

`vtss_vcap_u48_t vtss_vce_frame_snap_t::data`

Data

Definition at line 963 of file vtss_l2_api.h.

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

- `vtss_api/include/vtss_l2_api.h`

7.279 vtss_vce_key_t Struct Reference

VCE Key.

```
#include <vtss_l2_api.h>
```

Data Fields

- `BOOL port_list [VTSS_PORT_ARRAY_SIZE]`
- `vtss_vce_mac_t mac`
- `vtss_vce_tag_t tag`
- `vtss_vce_type_t type`
- union {
 - `vtss_vce_frame_etype_t etype`
 - `vtss_vce_frame_llc_t llc`
 - `vtss_vce_frame_snap_t snap`
 - `vtss_vce_frame_ipv4_t ipv4`
 - `vtss_vce_frame_ipv6_t ipv6`}

7.279.1 Detailed Description

VCE Key.

Definition at line 987 of file vtss_l2_api.h.

7.279.2 Field Documentation

7.279.2.1 port_list

```
BOOL vtss_vce_key_t::port_list[VTSS_PORT_ARRAY_SIZE]
```

Port list

Definition at line 989 of file vtss_l2_api.h.

7.279.2.2 mac

```
vtss_vce_mac_t vtss_vce_key_t::mac
```

MAC header

Definition at line 990 of file vtss_l2_api.h.

7.279.2.3 tag

`vtss_vce_tag_t` `vtss_vce_key_t::tag`

Tag

Definition at line 991 of file vtss_l2_api.h.

7.279.2.4 type

`vtss_vce_type_t` `vtss_vce_key_t::type`

VCE frame type

Definition at line 992 of file vtss_l2_api.h.

7.279.2.5 etype

`vtss_vce_frame_etype_t` `vtss_vce_key_t::etype`

VTSS_VCE_TYPE_ETYPE

Definition at line 997 of file vtss_l2_api.h.

7.279.2.6 llc

`vtss_vce_frame_llc_t` `vtss_vce_key_t::llc`

VTSS_VCE_TYPE_LLCC

Definition at line 998 of file vtss_l2_api.h.

7.279.2.7 snap

`vtss_vce_frame_snap_t` `vtss_vce_key_t::snap`

VTSS_VCE_TYPE_SNAP

Definition at line 999 of file vtss_l2_api.h.

7.279.2.8 ipv4

`vtss_vce_frame_ipv4_t vtss_vce_key_t::ipv4`

VTSS_VCE_TYPE_IPV4

Definition at line 1000 of file vtss_l2_api.h.

7.279.2.9 ipv6

`vtss_vce_frame_ipv6_t vtss_vce_key_t::ipv6`

VTSS_VCE_TYPE_IPV6

Definition at line 1001 of file vtss_l2_api.h.

7.279.2.10 frame

`union { ... } vtss_vce_key_t::frame`

Frame type specific data

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

- `vtss_api/include/vtss_l2_api.h`

7.280 vtss_vce_mac_t Struct Reference

VCE MAC header information.

```
#include <vtss_l2_api.h>
```

Data Fields

- `vtss_vcap_bit_t dmac_mc`
- `vtss_vcap_bit_t dmac_bc`
- `vtss_vcap_u48_t smac`

7.280.1 Detailed Description

VCE MAC header information.

Definition at line 930 of file vtss_l2_api.h.

7.280.2 Field Documentation

7.280.2.1 dmac_mc

`vtss_vcap_bit_t` `vtss_vce_mac_t::dmac_mc`

Multicast DMAC

Definition at line 932 of file `vtss_l2_api.h`.

7.280.2.2 dmac_bc

`vtss_vcap_bit_t` `vtss_vce_mac_t::dmac_bc`

Broadcast DMAC

Definition at line 933 of file `vtss_l2_api.h`.

7.280.2.3 smac

`vtss_vcap_u48_t` `vtss_vce_mac_t::smac`

SMAC

Definition at line 934 of file `vtss_l2_api.h`.

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

- `vtss_api/include/vtss_l2_api.h`

7.281 vtss_vce_t Struct Reference

VLAN Control Entry.

```
#include <vtss_l2_api.h>
```

Data Fields

- `vtss_vce_id_t id`
- `vtss_vce_key_t key`
- `vtss_vce_action_t action`

7.281.1 Detailed Description

VLAN Control Entry.

Definition at line 1013 of file vtss_l2_api.h.

7.281.2 Field Documentation

7.281.2.1 id

`vtss_vce_id_t` `vtss_vce_t::id`

VCE ID

Definition at line 1015 of file vtss_l2_api.h.

7.281.2.2 key

`vtss_vce_key_t` `vtss_vce_t::key`

VCE Key

Definition at line 1016 of file vtss_l2_api.h.

7.281.2.3 action

`vtss_vce_action_t` `vtss_vce_t::action`

VCE Action

Definition at line 1017 of file vtss_l2_api.h.

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

- vtss_api/include/[vtss_l2_api.h](#)

7.282 **vtss_vce_tag_t** Struct Reference

VCE tag information.

```
#include <vtss_l2_api.h>
```

Data Fields

- `vtss_vcap_vid_t vid`
- `vtss_vcap_u8_t pcp`
- `vtss_vcap_bit_t dei`
- `vtss_vcap_bit_t tagged`
- `vtss_vcap_bit_t s_tag`

7.282.1 Detailed Description

VCE tag information.

Definition at line 938 of file vtss_l2_api.h.

7.282.2 Field Documentation

7.282.2.1 vid

`vtss_vcap_vid_t vtss_vce_tag_t::vid`

VLAN ID (12 bit)

Definition at line 940 of file vtss_l2_api.h.

7.282.2.2 pcp

`vtss_vcap_u8_t vtss_vce_tag_t::pcp`

PCP (3 bit)

Definition at line 941 of file vtss_l2_api.h.

7.282.2.3 dei

`vtss_vcap_bit_t vtss_vce_tag_t::dei`

DEI

Definition at line 942 of file vtss_l2_api.h.

7.282.2.4 tagged

`vtss_vcap_bit_t` `vtss_vce_tag_t::tagged`

Tagged/untagged frame

Definition at line 943 of file `vtss_l2_api.h`.

7.282.2.5 s_tag

`vtss_vcap_bit_t` `vtss_vce_tag_t::s_tag`

S-tag type

Definition at line 944 of file `vtss_l2_api.h`.

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

- `vtss_api/include/vtss_l2_api.h`

7.283 vtss_vcl_port_conf_t Struct Reference

VCL port configuration.

```
#include <vtss_l2_api.h>
```

Data Fields

- `BOOL dmac_dip`

7.283.1 Detailed Description

VCL port configuration.

Definition at line 878 of file `vtss_l2_api.h`.

7.283.2 Field Documentation

7.283.2.1 dmac_dip

`BOOL vtss_vcl_port_conf_t::dmac_dip`

Enable DMAC/DIP matching (default SMAC/SIP)

Definition at line 879 of file `vtss_l2_api.h`.

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

- `vtss_api/include/vtss_l2_api.h`

7.284 vtss_vid_mac_t Struct Reference

MAC Address in specific VLAN.

```
#include <types.h>
```

Data Fields

- `vtss_vid_t vid`
- `vtss_mac_t mac`

7.284.1 Detailed Description

MAC Address in specific VLAN.

Definition at line 652 of file `types.h`.

7.284.2 Field Documentation

7.284.2.1 vid

`vtss_vid_t vtss_vid_mac_t::vid`

VLAN ID

Definition at line 654 of file `types.h`.

7.284.2.2 mac

`vtss_mac_t vtss_vid_mac_t::mac`

MAC address

Definition at line 655 of file types.h.

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

- [vtss_api/include/vtss/api/types.h](#)

7.285 vtss_vlan_conf_t Struct Reference

VLAN configuration.

```
#include <vtss_l2_api.h>
```

Data Fields

- [vtss_etype_t s_etype](#)

7.285.1 Detailed Description

VLAN configuration.

Definition at line 625 of file vtss_l2_api.h.

7.285.2 Field Documentation

7.285.2.1 s_etype

`vtss_etype_t vtss_vlan_conf_t::s_etype`

Alternative S-tag Ethernet Type

Definition at line 626 of file vtss_l2_api.h.

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

- [vtss_api/include/vtss_l2_api.h](#)

7.286 vtss_vlan_port_conf_t Struct Reference

VLAN port configuration.

```
#include <vtss_l2_api.h>
```

Data Fields

- `vtss_vlan_port_type_t port_type`
- `vtss_vid_t pvid`
- `vtss_vid_t untagged_vid`
- `vtss_vlan_frame_t frame_type`
- `BOOL ingress_filter`

7.286.1 Detailed Description

VLAN port configuration.

Definition at line 664 of file `vtss_l2_api.h`.

7.286.2 Field Documentation

7.286.2.1 port_type

```
vtss_vlan_port_type_t vtss_vlan_port_conf_t::port_type
```

Port type (ingress and egress)

Definition at line 671 of file `vtss_l2_api.h`.

7.286.2.2 pvid

```
vtss_vid_t vtss_vlan_port_conf_t::pvid
```

Port VLAN ID (PVID, ingress)

Definition at line 673 of file `vtss_l2_api.h`.

7.286.2.3 untagged_vid

`vtss_vid_t` `vtss_vlan_port_conf_t::untagged_vid`

Port untagged VLAN ID (UVID, egress)

Definition at line 674 of file vtss_l2_api.h.

7.286.2.4 frame_type

`vtss_vlan_frame_t` `vtss_vlan_port_conf_t::frame_type`

Acceptable frame type (ingress)

Definition at line 675 of file vtss_l2_api.h.

7.286.2.5 ingress_filter

`BOOL` `vtss_vlan_port_conf_t::ingress_filter`

Ingress filtering

Definition at line 676 of file vtss_l2_api.h.

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

- vtss_api/include/[vtss_l2_api.h](#)

7.287 vtss_vlan_tag_t Struct Reference

```
#include <types.h>
```

Data Fields

- [vtss_etype_t tpid](#)
- [vtss_tagprio_t pcp](#)
- `BOOL dei`
- `vtss_vid_t vid`

7.287.1 Detailed Description

VLAN tag with "arbitrary" TPID.

Definition at line 632 of file types.h.

7.287.2 Field Documentation

7.287.2.1 tpid

`vtss_etype_t` `vtss_vlan_tag_t::tpid`

Tag Protocol Identifier

Definition at line 633 of file types.h.

7.287.2.2 pcp

`vtss_tagprio_t` `vtss_vlan_tag_t::pcp`

Priority Code Point

Definition at line 634 of file types.h.

7.287.2.3 dei

`BOOL` `vtss_vlan_tag_t::dei`

Drop Eligible Indicator

Definition at line 635 of file types.h.

7.287.2.4 vid

`vtss_vid_t` `vtss_vlan_tag_t::vid`

VLAN Identifier

Definition at line 636 of file types.h.

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

- `vtss_api/include/vtss/api/types.h`

7.288 vtss_vlan_trans_grp2vlan_conf_t Struct Reference

VLAN translation group-to-VLAN configuration.

```
#include <vtss_l2_api.h>
```

Data Fields

- `u16 group_id`
- `vtss_vid_t vid`
- `vtss_vid_t trans_vid`

7.288.1 Detailed Description

VLAN translation group-to-VLAN configuration.

Definition at line 1104 of file `vtss_l2_api.h`.

7.288.2 Field Documentation

7.288.2.1 group_id

```
u16 vtss_vlan_trans_grp2vlan_conf_t::group_id
```

Group ID

Definition at line 1105 of file `vtss_l2_api.h`.

7.288.2.2 vid

```
vtss_vid_t vtss_vlan_trans_grp2vlan_conf_t::vid
```

VLAN ID

Definition at line 1106 of file `vtss_l2_api.h`.

7.288.2.3 trans_vid

`vtss_vid_t vtss_vlan_trans_grp2vlan_conf_t::trans_vid`

Translated VLAN ID

Definition at line 1107 of file `vtss_l2_api.h`.

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

- `vtss_api/include/vtss_l2_api.h`

7.289 vtss_vlan_trans_port2grp_conf_t Struct Reference

VLAN translation port-to-group configuration.

```
#include <vtss_l2_api.h>
```

Data Fields

- `u16 group_id`
- `u8 ports [VTSS_VLAN_TRANS_PORT_BF_SIZE]`

7.289.1 Detailed Description

VLAN translation port-to-group configuration.

Definition at line 1098 of file `vtss_l2_api.h`.

7.289.2 Field Documentation

7.289.2.1 group_id

`u16 vtss_vlan_trans_port2grp_conf_t::group_id`

Group ID

Definition at line 1099 of file `vtss_l2_api.h`.

7.289.2.2 ports

```
u8 vtss_vlan_trans_port2grp_conf_t::ports[VTSS_VLAN_TRANS_PORT_BF_SIZE]
```

Ports Bitfield

Definition at line 1100 of file vtss_l2_api.h.

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

- [vtss_api/include/vtss_l2_api.h](#)

7.290 vtss_vlan_vid_conf_t Struct Reference

VLAN ID configuration.

```
#include <vtss_l2_api.h>
```

Data Fields

- [BOOL learning](#)
- [BOOL mirror](#)
- [vtss_vid_t fid](#)

7.290.1 Detailed Description

VLAN ID configuration.

Definition at line 740 of file vtss_l2_api.h.

7.290.2 Field Documentation

7.290.2.1 learning

```
BOOL vtss_vlan_vid_conf_t::learning
```

Enable/disable learning

Definition at line 742 of file vtss_l2_api.h.

7.290.2.2 mirror

`BOOL vtss_vlan_vid_conf_t::mirror`

Enable/disable mirroring

Definition at line 743 of file `vtss_l2_api.h`.

7.290.2.3 fid

`vtss_vid_t vtss_vlan_vid_conf_t::fid`

Forwarding ID for SVL/IVL control

Definition at line 745 of file `vtss_l2_api.h`.

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

- `vtss_api/include/vtss_l2_api.h`

7.291 vtss_vstax_conf_t Struct Reference

VStaX configuration for switch.

```
#include <vtss_l2_api.h>
```

Data Fields

- `vtss_vstax_upsid_t upsid_0`
- `vtss_vstax_upsid_t upsid_1`
- `vtss_port_no_t port_0`
- `vtss_port_no_t port_1`
- `BOOL cmef_disable`

7.291.1 Detailed Description

VStaX configuration for switch.

Definition at line 2324 of file `vtss_l2_api.h`.

7.291.2 Field Documentation

7.291.2.1 upsid_0

`vtss_vstax_upsid_t vtss_vstax_conf_t::upsid_0`

Base UPSID of unit 0

Definition at line 2325 of file vtss_l2_api.h.

7.291.2.2 upsid_1

`vtss_vstax_upsid_t vtss_vstax_conf_t::upsid_1`

Base UPSID of unit 1 (if present)

Definition at line 2326 of file vtss_l2_api.h.

7.291.2.3 port_0

`vtss_port_no_t vtss_vstax_conf_t::port_0`

First stack port or VTSS_PORT_NO_NONE

Definition at line 2327 of file vtss_l2_api.h.

7.291.2.4 port_1

`vtss_port_no_t vtss_vstax_conf_t::port_1`

Second stack port or VTSS_PORT_NO_NONE

Definition at line 2328 of file vtss_l2_api.h.

7.291.2.5 cmef_disable

`BOOL vtss_vstax_conf_t::cmef_disable`

Disable Congestion Management

Definition at line 2330 of file vtss_l2_api.h.

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

- vtss_api/include/[vtss_l2_api.h](#)

7.292 vtss_vstax_glag_entry_t Struct Reference

GLAG info.

```
#include <vtss_l2_api.h>
```

Data Fields

- `vtss_vstax_upsid_t upsid`
- `vtss_vstax_upspn_t upspn`

7.292.1 Detailed Description

GLAG info.

Definition at line 1611 of file vtss_l2_api.h.

7.292.2 Field Documentation

7.292.2.1 upsid

```
vtss_vstax_upsid_t vtss_vstax_glag_entry_t::upsid
```

UPS identifier

Definition at line 1613 of file vtss_l2_api.h.

7.292.2.2 upspn

```
vtss_vstax_upspn_t vtss_vstax_glag_entry_t::upspn
```

Logical port on the UPS

Definition at line 1614 of file vtss_l2_api.h.

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

- vtss_api/include/[vtss_l2_api.h](#)

7.293 vtss_vstax_port_conf_t Struct Reference

VStaX setup for port.

```
#include <vtss_l2_api.h>
```

Data Fields

- `u8 ttl`
- `BOOL mirror`

7.293.1 Detailed Description

VStaX setup for port.

Definition at line 2358 of file `vtss_l2_api.h`.

7.293.2 Field Documentation

7.293.2.1 ttl

`u8 vtss_vstax_port_conf_t::ttl`

TTL, 0-31

Definition at line 2359 of file `vtss_l2_api.h`.

7.293.2.2 mirror

`BOOL vtss_vstax_port_conf_t::mirror`

Mirror port reachable via VStaX port

Definition at line 2360 of file `vtss_l2_api.h`.

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

- `vtss_api/include/vtss_l2_api.h`

7.294 vtss_vstax_route_entry_t Struct Reference

UPSID Route Entry.

```
#include <vtss_l2_api.h>
```

Data Fields

- `BOOL stack_port_a`
- `BOOL stack_port_b`

7.294.1 Detailed Description

UPSID Route Entry.

Definition at line 2425 of file vtss_l2_api.h.

7.294.2 Field Documentation

7.294.2.1 stack_port_a

`BOOL vtss_vstax_route_entry_t::stack_port_a`

UPSID is reachable through port A

Definition at line 2426 of file vtss_l2_api.h.

7.294.2.2 stack_port_b

`BOOL vtss_vstax_route_entry_t::stack_port_b`

UPSID is reachable through port B

Definition at line 2427 of file vtss_l2_api.h.

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

- vtss_api/include/[vtss_l2_api.h](#)

7.295 vtss_vstax_route_table_t Struct Reference

UPSID Route Table.

```
#include <vtss_l2_api.h>
```

Data Fields

- [vtss_vstax_topology_type_t topology_type](#)
- [vtss_vstax_route_entry_t table \[VTSS_VSTAX_UPSIDS\]](#)

7.295.1 Detailed Description

UPSID Route Table.

Definition at line 2431 of file vtss_l2_api.h.

7.295.2 Field Documentation

7.295.2.1 topology_type

`vtss_vstax_topology_type_t` `vtss_vstax_route_table_t::topology_type`

Topology type

Definition at line 2432 of file vtss_l2_api.h.

7.295.2.2 table

`vtss_vstax_route_entry_t` `vtss_vstax_route_table_t::table[VTSS_VSTAX_UPSIDS]`

UPSID is index into table

Definition at line 2433 of file vtss_l2_api.h.

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

- vtss_api/include/vtss_l2_api.h

7.296 vtss_vstax_rx_header_t Struct Reference

VStaX frame header used for reception.

```
#include <vtss_packet_api.h>
```

Data Fields

- `BOOL valid`
- `BOOL sp`
- `vtss_vstax_upsid_t upsid`
- `vtss_port_no_t port_no`
- `vtss_glag_no_t glag_no`
- `vtss_isdx_t isdx`

7.296.1 Detailed Description

VStaX frame header used for reception.

Definition at line 190 of file vtss_packet_api.h.

7.296.2 Field Documentation

7.296.2.1 valid

`BOOL vtss_vstax_rx_header_t::valid`

TRUE if frame was VStaX tagged

Definition at line 192 of file vtss_packet_api.h.

7.296.2.2 sp

`BOOL vtss_vstax_rx_header_t::sp`

TRUE if received on super-prio

Definition at line 193 of file vtss_packet_api.h.

7.296.2.3 upsid

`vtss_vstax_upsid_t vtss_vstax_rx_header_t::upsid`

Ingress Unit Port Set ID

Definition at line 194 of file vtss_packet_api.h.

7.296.2.4 port_no

`vtss_port_no_t vtss_vstax_rx_header_t::port_no`

Ingress Unit port number (or zero)

Definition at line 195 of file vtss_packet_api.h.

7.296.2.5 glag_no

`vtss_glag_no_t vtss_vstax_rx_header_t::glag_no`

Ingress GLAG (or zero)

Definition at line 196 of file vtss_packet_api.h.

7.296.2.6 isdx

`vtss_isdx_t vtss_vstax_rx_header_t::isdx`

12 bit ingress service index

Definition at line 198 of file vtss_packet_api.h.

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

- vtss_api/include/[vtss_packet_api.h](#)

7.297 vtss_vstax_tx_header_t Struct Reference

VStaX frame header used for transmission.

```
#include <vtss_packet_api.h>
```

Data Fields

- [vtss_vstax_fwd_mode_t fwd_mode](#)
- [vtss_vstax_ttl_t ttl](#)
- [vtss_prio_t prio](#)
- [vtss_vstax_upsid_t upsid](#)
- [vtss_tci_t tci](#)
- [vtss_port_no_t port_no](#)
- u8 chip_port
- [vtss_glag_no_t glag_no](#)
- [vtss_packet_rx_queue_t queue_no](#)
- BOOL keep_ttl
- u8 dp

7.297.1 Detailed Description

VStaX frame header used for transmission.

Definition at line 453 of file vtss_packet_api.h.

7.297.2 Field Documentation

7.297.2.1 fwd_mode

`vtss_vstax_fwd_mode_t` `vtss_vstax_tx_header_t::fwd_mode`

Frame forward mode

Definition at line 455 of file vtss_packet_api.h.

7.297.2.2 ttl

`vtss_vstax_ttl_t` `vtss_vstax_tx_header_t::ttl`

TTL value or VTSS_VSTAX_TTL_PORT

Definition at line 456 of file vtss_packet_api.h.

7.297.2.3 prio

`vtss_prio_t` `vtss_vstax_tx_header_t::prio`

Priority, VTSS_PRIO_SUPER allowed

Definition at line 457 of file vtss_packet_api.h.

7.297.2.4 upsid

`vtss_vstax_upsid_t` `vtss_vstax_tx_header_t::upsid`

Dest. unit port set: FWD_MODE_UPSID_PORT/CPU_UPSID

Definition at line 458 of file vtss_packet_api.h.

7.297.2.5 tci

`vtss_tci_t` `vtss_vstax_tx_header_t::tci`

VLAN tag information

Definition at line 459 of file vtss_packet_api.h.

7.297.2.6 port_no

`vtss_port_no_t vtss_vstax_tx_header_t::port_no`

Dest. port: FWD_MODE_UPSID_PORT - Src. port : FWD_MODE_CPU_UPSID/ALL

Definition at line 460 of file vtss_packet_api.h.

7.297.2.7 chip_port

`u8 vtss_vstax_tx_header_t::chip_port`

If port_no is VTSS_PORT_NO_NONE, then chip_port is used as is

Definition at line 461 of file vtss_packet_api.h.

7.297.2.8 glag_no

`vtss_glag_no_t vtss_vstax_tx_header_t::glag_no`

GLAG number: FWD_MODE_CPU_UPSID/ALL

Definition at line 463 of file vtss_packet_api.h.

7.297.2.9 queue_no

`vtss_packet_rx_queue_t vtss_vstax_tx_header_t::queue_no`

CPU queue : FWD_MODE_CPU_UPSID/ALL

Definition at line 465 of file vtss_packet_api.h.

7.297.2.10 keep_ttl

`BOOL vtss_vstax_tx_header_t::keep_ttl`

Special TTL handling : FWD_MODE_CPU_ALL

Definition at line 467 of file vtss_packet_api.h.

7.297.2.11 dp

`u8 vtss_vstax_tx_header_t::dp`

2 bits drop precedence level

Definition at line 468 of file `vtss_packet_api.h`.

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

- `vtss_api/include/vtss_packet_api.h`

7.298 vtss_wol_mac_addr_t Struct Reference

Structure for Wake-On-LAN MAC Address.

```
#include <vtss_phy_api.h>
```

Data Fields

- `u8 addr [MAX_WOL_MAC_ADDR_SIZE]`

7.298.1 Detailed Description

Structure for Wake-On-LAN MAC Address.

Definition at line 1656 of file `vtss_phy_api.h`.

7.298.2 Field Documentation

7.298.2.1 addr

`u8 vtss_wol_mac_addr_t::addr [MAX_WOL_MAC_ADDR_SIZE]`

Wake-On-LAN MAC Address - 6 bytes

Definition at line 1658 of file `vtss_phy_api.h`.

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

- `vtss_api/include/vtss_phy_api.h`

Chapter 8

File Documentation

8.1 vtss_api/include/vtss/api/l2_types.h File Reference

Layer 2 Public API Header for l2.

```
#include <vtss/api/types.h>
```

Data Structures

- struct `vtss_aggr_mode_t`
Aggregation traffic distribution mode.

Enumerations

- enum `vtss_sfflow_type_t`{ `VTSS_SFLOW_TYPE_NONE` = 0, `VTSS_SFLOW_TYPE_RX`, `VTSS_SFLOW_TYPE_TX`,
`VTSS_SFLOW_TYPE_ALL` }
sFlow sampler type.

8.1.1 Detailed Description

Layer 2 Public API Header for l2.

This header file describes public Layer 2 datatypes

8.1.2 Enumeration Type Documentation

8.1.2.1 vtss_sfflow_type_t

```
enum vtss_sfflow_type_t
```

sFlow sampler type.

The API supports sampling ingress and egress separately, as well as simultaneously.

Enumerator

VTSS_SFLOW_TYPE_NONE	Sampler is not enabled on the port.
VTSS_SFLOW_TYPE_RX	Sampler is enabled for ingress on the port.
VTSS_SFLOW_TYPE_TX	Sampler is enabled for egress on the port.
VTSS_SFLOW_TYPE_ALL	Sampler is enabled for both ingress and egress on the port.

Definition at line 53 of file l2_types.h.

8.2 vtss_api/include/vtss/api/options.h File Reference

Features and options.

Macros

- #define VTSS_ARCH_JAGUAR_1
- #define VTSS_ARCH_JAGUAR_1_CE_SWITCH
- #define VTSS_FEATURE_EVC
- #define VTSS_FEATURE_E_TREE
- #define VTSS_FEATURE_TIMESTAMP
- #define VTSS_FEATURE_PHY_TIMESTAMP
- #define VTSS_FEATURE_VSTAX
- #define VTSS_FEATURE_AGGR_GLAG
- #define VTSS_FEATURE_VSTAX_V2
- #define VTSS_PHY_TS_SPI_CLK_THRU_PPS0
- #define VTSS_FEATURE_FAN
- #define VTSS_FEATURE_PVLAN
- #define VTSS_OPT_TS_SPI_FPGA
- #define VTSS_CHIP_10G_PHY
- #define VTSS_FEATURE_MISC
- #define VTSS_FEATURE_SERIAL_GPIO
- #define VTSS_FEATURE_PORT_CONTROL
- #define VTSS_FEATURE_PORT_IFH
- #define VTSS_FEATURE_CLAUSE_37
- #define VTSS_FEATURE_10G
- #define VTSS_FEATURE_NPI
- #define VTSS_FEATURE_EXC_COL_CONT
- #define VTSS_FEATURE_PORT_CNT_ETHER_LIKE
- #define VTSS_FEATURE_PORT_CNT_BRIDGE
- #define VTSS_FEATURE_QOS
- #define VTSS_FEATURE_QCL
- #define VTSS_FEATURE_QCL_V2
- #define VTSS_FEATURE_QOS_PORT_POLICER_EXT
- #define VTSS_FEATURE_QOS_PORT_POLICER_EXT_FPS
- #define VTSS_FEATURE_QOS_PORT_POLICER_EXT_FC
- #define VTSS_FEATURE_QOS_PORT_POLICER_EXT_DPBL
- #define VTSS_FEATURE_QOS_PORT_POLICER_EXT_TTM
- #define VTSS_FEATURE_QOS_QUEUE_POLICER
- #define VTSS_FEATURE_QOS_QUEUE_TX
- #define VTSS_FEATURE_QOS_SCHEDULER_V2

- #define VTSS_FEATURE_QOS_TAG_REMARK_V2
- #define VTSS_FEATURE_QOS_CLASSIFICATION_V2
- #define VTSS_FEATURE_QOS_EGRESS_QUEUE_SHAPERS
- #define VTSS_FEATURE_QOS_EGRESS_QUEUE_SHAPERS_EB
- #define VTSS_FEATURE_QOS_DSCP_REMARK
- #define VTSS_FEATURE_QOS_DSCP_REMARK_V2
- #define VTSS_FEATURE_QOS_WRED
- #define VTSS_FEATURE_PACKET
- #define VTSS_FEATURE_PACKET_TX
- #define VTSS_FEATURE_PACKET_RX
- #define VTSS_FEATURE_PACKET_GROUPING
- #define VTSS_FEATURE_PACKET_PORT_REG
- #define VTSS_FEATURE_LAYER2
- #define VTSS_FEATURE_VLAN_PORT_V2
- #define VTSS_FEATURE_VLAN_TX_TAG
- #define VTSS_FEATURE_VLAN_SVL
- #define VTSS_FEATURE_MAC_AGE_AUTO
- #define VTSS_FEATURE_MAC_CPU_QUEUE
- #define VTSS_FEATURE_LAYER3
- #define VTSS_FEATURE_PORT_MUX
- #define VTSS_FEATURE_VCAP
- #define VTSS_FEATURE_ACL
- #define VTSS_FEATURE_ACL_V1
- #define VTSS_FEATURE_VCL
- #define VTSS_FEATURE_SYNCE
- #define VTSS_FEATURE_FAN
- #define VTSS_FEATURE_EEE
- #define VTSS_FEATURE_SERDES_MACRO_SETTINGS
- #define VTSS_FEATURE_LED_POW_REDUC
- #define VTSS_FEATURE_10GBASE_KR
- #define VTSS_FEATURE_IRQ_CONTROL
- #define VTSS_FEATURE_VLAN_TRANSLATION
- #define VTSS_FEATURE_SFLOW
- #define VTSS_PHY_10G_FIFO_SYNC
- #define VIPER_B_FIFO_RESET
- #define VTSS_FEATURE_QOS_POLICER_DLBB
- #define VTSS_CHIP CU PHY
- #define VTSS_OPT_TRACE 1
- #define VTSS_OPT_VAUI_EQ_CTRL 6
- #define VTSS_PHY_OPT_VERIPHYSY 1
- #define VTSS_FEATURE_WARM_START
- #define VTSS_FEATURE_SYNCE_10G
- #define VTSS_FEATURE_EDC_FW_LOAD
- #define VTSS_FEATURE_WIS
- #define VTSS_FEATURE_WARM_START
- #define VTSS_ARCH_MALIBU
- #define VTSS_ARCH_MALIBU_B
- #define VTSS_ARCH_VENICE_C
- #define VTSS_FEATURE_VCAP

8.2.1 Detailed Description

Features and options.

This header file describes target features and compile-time options

8.2.2 Macro Definition Documentation

8.2.2.1 VTSS_ARCH_JAGUAR_1

```
#define VTSS_ARCH_JAGUAR_1  
< Jaguar-1 CE switches Jaguar-1 architecture  
Definition at line 76 of file options.h.
```

8.2.2.2 VTSS_ARCH_JAGUAR_1_CE_SWITCH

```
#define VTSS_ARCH_JAGUAR_1_CE_SWITCH  
Jaguar-1 CE switch family  
Definition at line 77 of file options.h.
```

8.2.2.3 VTSS_FEATURE_EVC

```
#define VTSS_FEATURE_EVC  
Ethernet Virtual Connections  
Definition at line 78 of file options.h.
```

8.2.2.4 VTSS_FEATURE_E_TREE

```
#define VTSS_FEATURE_E_TREE  
EVC E-Tree  
Definition at line 79 of file options.h.
```

8.2.2.5 VTSS_FEATURE_TIMESTAMP

```
#define VTSS_FEATURE_TIMESTAMP  
Packet timestamp feature (for PTP/OAM)  
Definition at line 80 of file options.h.
```

8.2.2.6 VTSS_FEATURE_PHY_TIMESTAMP

```
#define VTSS_FEATURE_PHY_TIMESTAMP
```

PHY timestamp feature (for PTP/OAM)

Definition at line 81 of file options.h.

8.2.2.7 VTSS_FEATURE_VSTAX

```
#define VTSS_FEATURE_VSTAX
```

VStaX stacking

Definition at line 83 of file options.h.

8.2.2.8 VTSS_FEATURE_AGGR_GLAG

```
#define VTSS_FEATURE_AGGR_GLAG
```

Global link aggregations across stack

Definition at line 84 of file options.h.

8.2.2.9 VTSS_FEATURE_VSTAX_V2

```
#define VTSS_FEATURE_VSTAX_V2
```

VStaX stacking, as implemented on Jaguar1 (VStaX2/AF)

Definition at line 85 of file options.h.

8.2.2.10 VTSS_PHY_TS_SPI_CLK_THRU_PPS0

```
#define VTSS_PHY_TS_SPI_CLK_THRU_PPS0
```

Use 1588_PPS0 as New SPI_CLK, applicable only to 8574-15; old SPI_CLK pin will not be used anymore

Definition at line 87 of file options.h.

8.2.2.11 VTSS_FEATURE_FAN [1/2]

```
#define VTSS_FEATURE_FAN
```

Fan control

Definition at line 169 of file options.h.

8.2.2.12 VTSS_FEATURE_PVLAN

```
#define VTSS_FEATURE_PVLAN
```

< Jaguar-1 single chip switches Private VLANs

Definition at line 106 of file options.h.

8.2.2.13 VTSS_OPT_TS_SPI_FPGA

```
#define VTSS_OPT_TS_SPI_FPGA
```

< Jaguar-1 family

Definition at line 121 of file options.h.

8.2.2.14 VTSS_CHIP_10G_PHY

```
#define VTSS_CHIP_10G_PHY
```

10Gb 848x Phy API

Definition at line 122 of file options.h.

8.2.2.15 VTSS_FEATURE_MISC

```
#define VTSS_FEATURE_MISC
```

Miscellaneous

Definition at line 123 of file options.h.

8.2.2.16 VTSS FEATURE SERIAL GPIO

```
#define VTSS_FEATURE_SERIAL_GPIO
```

Serial GPIO control

Definition at line 124 of file options.h.

8.2.2.17 VTSS FEATURE PORT CONTROL

```
#define VTSS_FEATURE_PORT_CONTROL
```

Port control

Definition at line 125 of file options.h.

8.2.2.18 VTSS FEATURE PORT IFH

```
#define VTSS_FEATURE_PORT_IFH
```

Port IFH control

Definition at line 126 of file options.h.

8.2.2.19 VTSS FEATURE CLAUSE_37

```
#define VTSS_FEATURE_CLAUSE_37
```

IEEE 802.3 clause 37 auto-negotiation

Definition at line 127 of file options.h.

8.2.2.20 VTSS FEATURE_10G

```
#define VTSS_FEATURE_10G
```

10G ports

Definition at line 128 of file options.h.

8.2.2.21 VTSS_FEATURE_NPI

```
#define VTSS_FEATURE_NPI
```

NPI port

Definition at line 129 of file options.h.

8.2.2.22 VTSS_FEATURE_EXC_COL_CONT

```
#define VTSS_FEATURE_EXC_COL_CONT
```

Excessive collision continuation

Definition at line 130 of file options.h.

8.2.2.23 VTSS_FEATURE_PORT_CNT_ETHER_LIKE

```
#define VTSS_FEATURE_PORT_CNT_ETHER_LIKE
```

Ethernet-like counters

Definition at line 131 of file options.h.

8.2.2.24 VTSS_FEATURE_PORT_CNT_BRIDGE

```
#define VTSS_FEATURE_PORT_CNT_BRIDGE
```

Bridge counters

Definition at line 132 of file options.h.

8.2.2.25 VTSS_FEATURE_QOS

```
#define VTSS_FEATURE_QOS
```

QoS

Definition at line 133 of file options.h.

8.2.2.26 VTSS_FEATURE_QCL

```
#define VTSS_FEATURE_QCL
```

QoS: QoS Control Lists

Definition at line 134 of file options.h.

8.2.2.27 VTSS_FEATURE_QCL_V2

```
#define VTSS_FEATURE_QCL_V2
```

QoS: QoS Control Lists, V2 features

Definition at line 135 of file options.h.

8.2.2.28 VTSS_FEATURE_QOS_PORT_POLICER_EXT

```
#define VTSS_FEATURE_QOS_PORT_POLICER_EXT
```

QoS: Port Policer Extensions

Definition at line 136 of file options.h.

8.2.2.29 VTSS_FEATURE_QOS_PORT_POLICER_EXT_FPS

```
#define VTSS_FEATURE_QOS_PORT_POLICER_EXT_FPS
```

QoS: Port Policer has frame rate support

Definition at line 137 of file options.h.

8.2.2.30 VTSS_FEATURE_QOS_PORT_POLICER_EXT_FC

```
#define VTSS_FEATURE_QOS_PORT_POLICER_EXT_FC
```

QoS: Port Policer has flow control support

Definition at line 138 of file options.h.

8.2.2.31 VTSS FEATURE QOS PORT POLICER EXT DPBL

```
#define VTSS_FEATURE_QOS_PORT_POLICER_EXT_DPBL
```

QoS: Port Policer has Drop Precedence Bypass Level support

Definition at line 139 of file options.h.

8.2.2.32 VTSS FEATURE QOS PORT POLICER EXT TTM

```
#define VTSS_FEATURE_QOS_PORT_POLICER_EXT_TTM
```

QoS: Port Policer has Traffic_Type Mask support

Definition at line 140 of file options.h.

8.2.2.33 VTSS FEATURE QOS QUEUE POLICER

```
#define VTSS_FEATURE_QOS_QUEUE_POLICER
```

QoS: Has Ingress Queue Policers

Definition at line 141 of file options.h.

8.2.2.34 VTSS FEATURE QOS QUEUE TX

```
#define VTSS_FEATURE_QOS_QUEUE_TX
```

QoS: Has TX Queue support

Definition at line 142 of file options.h.

8.2.2.35 VTSS FEATURE QOS SCHEDULER V2

```
#define VTSS_FEATURE_QOS_SCHEDULER_V2
```

QoS: 2. version of scheduler

Definition at line 143 of file options.h.

8.2.2.36 VTSS_FEATURE_QOS_TAG_REMARK_V2

```
#define VTSS_FEATURE_QOS_TAG_REMARK_V2
```

QoS: 2. version of tag priority remarking

Definition at line 144 of file options.h.

8.2.2.37 VTSS_FEATURE_QOS_CLASSIFICATION_V2

```
#define VTSS_FEATURE_QOS_CLASSIFICATION_V2
```

QoS: 2. version of classification

Definition at line 145 of file options.h.

8.2.2.38 VTSS_FEATURE_QOS_EGRESS_QUEUE_SHAPERS

```
#define VTSS_FEATURE_QOS_EGRESS_QUEUE_SHAPERS
```

QoS: Has Egress Queue Shapers

Definition at line 146 of file options.h.

8.2.2.39 VTSS_FEATURE_QOS_EGRESS_QUEUE_SHAPERS_EB

```
#define VTSS_FEATURE_QOS_EGRESS_QUEUE_SHAPERS_EB
```

QoS: Egress Queue Shapers has Excess Bandwidth support

Definition at line 147 of file options.h.

8.2.2.40 VTSS_FEATURE_QOS_DSCP_REMARK

```
#define VTSS_FEATURE_QOS_DSCP_REMARK
```

QoS: Has DSCP remarking

Definition at line 148 of file options.h.

8.2.2.41 VTSS FEATURE QOS_DSCP_REMARK_V2

```
#define VTSS_FEATURE_QOS_DSCP_REMARK_V2
```

QoS: 2. version of DSCP remarking

Definition at line 149 of file options.h.

8.2.2.42 VTSS FEATURE QOS_WRED

```
#define VTSS_FEATURE_QOS_WRED
```

QoS: WRED pr. port

Definition at line 150 of file options.h.

8.2.2.43 VTSS FEATURE PACKET

```
#define VTSS_FEATURE_PACKET
```

CPU Rx/Tx frame configuration

Definition at line 151 of file options.h.

8.2.2.44 VTSS FEATURE PACKET_TX

```
#define VTSS_FEATURE_PACKET_TX
```

CPU Tx frame

Definition at line 152 of file options.h.

8.2.2.45 VTSS FEATURE PACKET_RX

```
#define VTSS_FEATURE_PACKET_RX
```

CPU Rx frame

Definition at line 153 of file options.h.

8.2.2.46 VTSS_FEATURE_PACKET_GROUPING

```
#define VTSS_FEATURE_PACKET_GROUPING
```

Extraction and injection occurs through extraction and injection groups rather than queues.

Definition at line 154 of file options.h.

8.2.2.47 VTSS_FEATURE_PACKET_PORT_REG

```
#define VTSS_FEATURE_PACKET_PORT_REG
```

Packet registration per port

Definition at line 155 of file options.h.

8.2.2.48 VTSS_FEATURE_LAYER2

```
#define VTSS_FEATURE_LAYER2
```

Layer 2 (switching)

Definition at line 156 of file options.h.

8.2.2.49 VTSS_FEATURE_VLAN_PORT_V2

```
#define VTSS_FEATURE_VLAN_PORT_V2
```

VLAN port configuration, V2 features

Definition at line 157 of file options.h.

8.2.2.50 VTSS_FEATURE_VLAN_TX_TAG

```
#define VTSS_FEATURE_VLAN_TX_TAG
```

VLAN tagging per (VID, port)

Definition at line 158 of file options.h.

8.2.2.51 VTSS_FEATURE_VLAN_SVL

```
#define VTSS_FEATURE_VLAN_SVL
```

Shared VLAN Learning

Definition at line 159 of file options.h.

8.2.2.52 VTSS_FEATURE_MAC_AGE_AUTO

```
#define VTSS_FEATURE_MAC_AGE_AUTO
```

Automatic MAC address ageing

Definition at line 160 of file options.h.

8.2.2.53 VTSS_FEATURE_MAC_CPU_QUEUE

```
#define VTSS_FEATURE_MAC_CPU_QUEUE
```

CPU queue per MAC address

Definition at line 161 of file options.h.

8.2.2.54 VTSS_FEATURE_LAYER3

```
#define VTSS_FEATURE_LAYER3
```

Layer 3 (routing)

Definition at line 162 of file options.h.

8.2.2.55 VTSS_FEATURE_PORT_MUX

```
#define VTSS_FEATURE_PORT_MUX
```

Port mux between serdes blocks and ports

Definition at line 163 of file options.h.

8.2.2.56 VTSS_FEATURE_VCAP [1/2]

```
#define VTSS_FEATURE_VCAP
```

VCAP

Definition at line 637 of file options.h.

8.2.2.57 VTSS_FEATURE_ACL

```
#define VTSS_FEATURE_ACL
```

Access Control Lists

Definition at line 165 of file options.h.

8.2.2.58 VTSS_FEATURE_ACL_V1

```
#define VTSS_FEATURE_ACL_V1
```

Access Control Lists, V2 features

Definition at line 166 of file options.h.

8.2.2.59 VTSS_FEATURE_VCL

```
#define VTSS_FEATURE_VCL
```

VLAN Control Lists

Definition at line 167 of file options.h.

8.2.2.60 VTSS_FEATURE_SYNCE

```
#define VTSS_FEATURE_SYNCE
```

SYNCE - L1 synchronization feature

Definition at line 168 of file options.h.

8.2.2.61 VTSS_FEATURE_FAN [2/2]

```
#define VTSS_FEATURE_FAN
```

Fan control

Definition at line 169 of file options.h.

8.2.2.62 VTSS_FEATURE_EEE

```
#define VTSS_FEATURE_EEE
```

Energy Efficient Ethernet

Definition at line 170 of file options.h.

8.2.2.63 VTSS_FEATURE_SERDES_MACRO_SETTINGS

```
#define VTSS_FEATURE_SERDES_MACRO_SETTINGS
```

Hooks for Serdes Macro configuration

Definition at line 171 of file options.h.

8.2.2.64 VTSS_FEATURE_LED_POW_REDUC

```
#define VTSS_FEATURE_LED_POW_REDUC
```

LED power reduction

Definition at line 172 of file options.h.

8.2.2.65 VTSS_FEATURE_10GBASE_KR

```
#define VTSS_FEATURE_10GBASE_KR
```

10GBASE_KR transmit equalization support IEEE802.3 Clause 72.7

Definition at line 173 of file options.h.

8.2.2.66 VTSS_FEATURE_IRQ_CONTROL

```
#define VTSS_FEATURE_IRQ_CONTROL
```

General IRQ support

Definition at line 174 of file options.h.

8.2.2.67 VTSS_FEATURE_VLAN_TRANSLATION

```
#define VTSS_FEATURE_VLAN_TRANSLATION
```

VLAN Translation

Definition at line 184 of file options.h.

8.2.2.68 VTSS_FEATURE_SFLOW

```
#define VTSS_FEATURE_SFLOW
```

Statistical flow sampling

Definition at line 185 of file options.h.

8.2.2.69 VTSS_PHY_10G_FIFO_SYNC

```
#define VTSS_PHY_10G_FIFO_SYNC
```

TSFIFO SYNC For 10G PHY

Definition at line 186 of file options.h.

8.2.2.70 VIPER_B_FIFO_RESET

```
#define VIPER_B_FIFO_RESET
```

Viper B 1588 FIFO sync

Definition at line 187 of file options.h.

8.2.2.71 VTSS_FEATURE_QOS_POLICER_DLDB

```
#define VTSS_FEATURE_QOS_POLICER_DLDB
```

DLB policers

Definition at line 328 of file options.h.

8.2.2.72 VTSS_CHIP CU PHY

```
#define VTSS_CHIP CU PHY
```

Copper PHY chip

Definition at line 540 of file options.h.

8.2.2.73 VTSS_OPT_TRACE

```
#define VTSS_OPT_TRACE 1
```

Trace enabled by default

Definition at line 548 of file options.h.

8.2.2.74 VTSS_OPT_VAUI_EQ_CTRL

```
#define VTSS_OPT_VAUI_EQ_CTRL 6
```

Default equalization control

Definition at line 605 of file options.h.

8.2.2.75 VTSS_PHY_OPT_VERIPHY

```
#define VTSS_PHY_OPT_VERIPHY 1
```

VeriPHY enabled by default

Definition at line 613 of file options.h.

8.2.2.76 VTSS_FEATURE_WARM_START [1/2]

```
#define VTSS_FEATURE_WARM_START
```

Warm start

Definition at line 624 of file options.h.

8.2.2.77 VTSS_FEATURE_SYNCE_10G

```
#define VTSS_FEATURE_SYNCE_10G
```

SYNCE - L1 synchronization feature for 10G PHYs

Definition at line 621 of file options.h.

8.2.2.78 VTSS_FEATURE_EDC_FW_LOAD

```
#define VTSS_FEATURE_EDC_FW_LOAD
```

848x EDC firmware will get loaded at initialization

Definition at line 622 of file options.h.

8.2.2.79 VTSS_FEATURE_WIS

```
#define VTSS_FEATURE_WIS
```

WAN interface sublayer functionality

Definition at line 623 of file options.h.

8.2.2.80 VTSS_FEATURE_WARM_START [2/2]

```
#define VTSS_FEATURE_WARM_START
```

Warm start

Definition at line 624 of file options.h.

8.2.2.81 VTSS_ARCH_MALIBU

```
#define VTSS_ARCH_MALIBU
```

Used for Malibu-A PHY

Definition at line 625 of file options.h.

8.2.2.82 VTSS_ARCH_MALIBU_B

```
#define VTSS_ARCH_MALIBU_B
```

Used for Malibu-B PHY

Definition at line 626 of file options.h.

8.2.2.83 VTSS_ARCH_VENICE_C

```
#define VTSS_ARCH_VENICE_C
```

Used for Venice-C PHY

Definition at line 627 of file options.h.

8.2.2.84 VTSS_FEATURE_VCAP [2/2]

```
#define VTSS_FEATURE_VCAP
```

VCAP

Definition at line 637 of file options.h.

8.3 vtss_api/include/vtss/api/phy.h File Reference

PHY Public API Header.

```
#include <vtss/api/types.h>
```

Macros

- #define VTSS_PHY_POWER_ACTIPHYS_BIT 0
- #define VTSS_PHY_POWER_DYNAMIC_BIT 1

Enumerations

- enum `vtss_phy_power_mode_t` { `VTSS_PHY_POWER_NOMINAL` = 0, `VTSS_PHY_POWER_ACTIPHY` = 1 << `VTSS_PHY_POWER_ACTIPHY_BIT`, `VTSS_PHY_POWER_DYNAMIC` = 1 << `VTSS_PHY_POWER_DYNAMIC_BIT`, `VTSS_PHY_POWER_ENABLED` = `VTSS_PHY_POWER_ACTIPHY` + `VTSS_PHY_POWER_DYNAMIC` }
- PHY power reduction modes.*
- enum `vtss_phy_veriphy_status_t` { `VTSS_VERIPHYS_STATUS_OK` = 0, `VTSS_VERIPHYS_STATUS_OPEN` = 1, `VTSS_VERIPHYS_STATUS_SHORT` = 2, `VTSS_VERIPHYS_STATUS_ABNORM` = 4, `VTSS_VERIPHYS_STATUS_SHORT_A` = 8, `VTSS_VERIPHYS_STATUS_SHORT_B` = 9, `VTSS_VERIPHYS_STATUS_SHORT_C` = 10, `VTSS_VERIPHYS_STATUS_SHORT_D` = 11, `VTSS_VERIPHYS_STATUS_COUPL_A` = 12, `VTSS_VERIPHYS_STATUS_COUPL_B` = 13, `VTSS_VERIPHYS_STATUS_COUPL_C` = 14, `VTSS_VERIPHYS_STATUS_COUPL_D` = 15, `VTSS_VERIPHYS_STATUS_UNKNOWN` = 16, `VTSS_VERIPHYS_STATUS_RUNNING` = 17 }
- VeriPHY status.*

8.3.1 Detailed Description

PHY Public API Header.

This header file describes public PHY data-types

8.3.2 Macro Definition Documentation

8.3.2.1 VTSS_PHY_POWER_ACTIPHY_BIT

```
#define VTSS_PHY_POWER_ACTIPHY_BIT 0
```

Defines the bit used to signaling that ActiPhy is enabled

Definition at line 42 of file phy.h.

8.3.2.2 VTSS_PHY_POWER_DYNAMIC_BIT

```
#define VTSS_PHY_POWER_DYNAMIC_BIT 1
```

Defines the bit used to signaling that PerfectReach is enabled

Definition at line 43 of file phy.h.

8.3.3 Enumeration Type Documentation

8.3.3.1 vtss_phy_power_mode_t

```
enum vtss_phy_power_mode_t
```

PHY power reduction modes.

Enumerator

VTSS_PHY_POWER_NOMINAL	Default power settings
VTSS_PHY_POWER_ACTIPHY	ActiPHY - Link down power savings enabled (Bit 0)
VTSS_PHY_POWER_DYNAMIC	PerfectReach - Link up power savings enabled (Bit 1)
VTSS_PHY_POWER_ENABLED	ActiPHY + PerfectReach enabled

Definition at line 46 of file phy.h.

8.3.3.2 vtss_phy_veriphy_status_t

```
enum vtss_phy_veriphy_status_t
```

VeriPHY status.

Enumerator

VTSS_VERIPHYS_STATUS_OK	Correctly terminated pair
VTSS_VERIPHYS_STATUS_OPEN	Open pair
VTSS_VERIPHYS_STATUS_SHORT	Short pair
VTSS_VERIPHYS_STATUS_ABNORM	Abnormal termination
VTSS_VERIPHYS_STATUS_SHORT_A	Cross-pair short to pair A
VTSS_VERIPHYS_STATUS_SHORT_B	Cross-pair short to pair B
VTSS_VERIPHYS_STATUS_SHORT_C	Cross-pair short to pair C
VTSS_VERIPHYS_STATUS_SHORT_D	Cross-pair short to pair D
VTSS_VERIPHYS_STATUS_COUPL_A	Abnormal cross-pair coupling, pair A
VTSS_VERIPHYS_STATUS_COUPL_B	Abnormal cross-pair coupling, pair B
VTSS_VERIPHYS_STATUS_COUPL_C	Abnormal cross-pair coupling, pair C
VTSS_VERIPHYS_STATUS_COUPL_D	Abnormal cross-pair coupling, pair D
VTSS_VERIPHYS_STATUS_UNKNOWN	Unknown - VeriPhy never started ?
VTSS_VERIPHYS_STATUS_RUNNING	VeriPhy is still running - No result yet

Definition at line 54 of file phy.h.

8.4 vtss_api/include/vtss/api/port.h File Reference

Port Public API Header.

```
#include <vtss/api/types.h>
#include <vtss/api/phy.h>
```

Data Structures

- struct `vtss_port_rmon_counters_t`
RMON counter structure (RFC 2819)
- struct `vtss_port_if_group_counters_t`
Interfaces Group counter structure (RFC 2863)
- struct `vtss_port_ethernet_like_counters_t`
Ethernet-like Interface counter structure (RFC 3635)
- struct `vtss_port_bridge_counters_t`
Port bridge counter structure (RFC 4188)
- struct `vtss_port_proprietary_counters_t`
Port proprietary counter structure.
- struct `vtss_port_counters_t`
Port counter structure.
- struct `port_custom_conf_t`
Port configuration.
- struct `vtss_port_status_t`
Port status parameter struct.

Macros

- `#define PORT_CAP_NONE 0x00000000`
- `#define PORT_CAP_AUTONEG 0x00000001`
- `#define PORT_CAP_10M_HDX 0x00000002`
- `#define PORT_CAP_10M_FDX 0x00000004`
- `#define PORT_CAP_100M_HDX 0x00000008`
- `#define PORT_CAP_100M_FDX 0x00000010`
- `#define PORT_CAP_1G_FDX 0x00000020`
- `#define PORT_CAP_2_5G_FDX 0x00000040`
- `#define PORT_CAP_5G_FDX 0x00000080`
- `#define PORT_CAP_10G_FDX 0x00000100`
- `#define PORT_CAP_FLOW_CTRL 0x00001000`
- `#define PORT_CAP_COPPER 0x00002000`
- `#define PORT_CAP_FIBER 0x00004000`
- `#define PORT_CAP_DUAL_COPPER 0x00008000`
- `#define PORT_CAP_DUAL_FIBER 0x00010000`
- `#define PORT_CAP_SD_ENABLE 0x00020000`
- `#define PORT_CAP_SD_HIGH 0x00040000`
- `#define PORT_CAP_SD_INTERNAL 0x00080000`
- `#define PORT_CAP_DUAL_FIBER_100FX 0x00100000`
- `#define PORT_CAP_XAUI_LANE_FLIP 0x00200000`
- `#define PORT_CAP_VTSS_10G_PHY 0x00400000`
- `#define PORT_CAP_SFP_DETECT 0x00800000`
- `#define PORT_CAP_STACKING 0x01000000`
- `#define PORT_CAP_DUAL_SFP_DETECT 0x02000000`
- `#define PORT_CAP_SFP_ONLY 0x04000000`
- `#define PORT_CAP_DUAL_COPPER_100FX 0x08000000`
- `#define PORT_CAP_HDX (PORT_CAP_10M_HDX | PORT_CAP_100M_HDX)`
- `#define PORT_CAP_TRI_SPEED_FDX (PORT_CAP_AUTONEG | PORT_CAP_1G_FDX | PORT_CAP_100M_FDX | PORT_CAP_10M_FDX | PORT_CAP_FLOW_CTRL)`
- `#define PORT_CAP_TRI_SPEED (PORT_CAP_TRI_SPEED_FDX | PORT_CAP_HDX)`
- `#define PORT_CAP_1G_PHY (PORT_CAP_COPPER | PORT_CAP_FIBER | PORT_CAP_DUAL_COPPER | PORT_CAP_DUAL_FIBER | PORT_CAP_DUAL_FIBER_100FX)`

- #define PORT_CAP_TRI_SPEED_COPPER (PORT_CAP_TRI_SPEED | PORT_CAP_COPPER)
- #define PORT_CAP_TRI_SPEED_FIBER (PORT_CAP_TRI_SPEED | PORT_CAP_FIBER)
- #define PORT_CAP_TRI_SPEED_DUAL_COPPER (PORT_CAP_TRI_SPEED | PORT_CAP_DUAL_COPPER)
- #define PORT_CAP_TRI_SPEED_DUAL_FIBER (PORT_CAP_TRI_SPEED | PORT_CAP_DUAL_FIBER)
- #define PORT_CAP_TRI_SPEED_DUAL_FIBER_100FX (PORT_CAP_TRI_SPEED | PORT_CAP_DUAL_FIBER_100FX)
- #define PORT_CAP_ANY_FIBER (PORT_CAP_FIBER | PORT_CAP_DUAL_FIBER_100FX | PORT_CAP_DUAL_FIBER | PORT_CAP_DUAL_COPPER | PORT_CAP_SFP_DETECT)
- #define PORT_CAP_SPEED_DUAL_ANY_FIBER_FIXED_SPEED (PORT_CAP_DUAL_FIBER_100FX | PORT_CAP_DUAL_FIBER | PORT_CAP_DUAL_COPPER)
- #define PORT_CAP_SPEED_DUAL_ANY_FIBER (PORT_CAP_DUAL_COPPER | PORT_CAP_DUAL_FIBER | PORT_CAP_DUAL_FIBER_100FX | PORT_CAP_DUAL_SFP_DETECT)
- #define PORT_CAP_TRI_SPEED_DUAL_ANY_FIBER (PORT_CAP_TRI_SPEED | PORT_CAP_SPEED_DUAL_ANY_FIBER)
- #define PORT_CAP_TRI_SPEED_DUAL_ANY_FIBER_FIXED_SFP_SPEED (PORT_CAP_TRI_SPEED | PORT_CAP_SPEED_DUAL_ANY_FIBER_FIXED_SPEED)
- #define PORT_CAP_DUAL_FIBER_1000X (PORT_CAP_DUAL_FIBER | PORT_CAP_DUAL_COPPER)
- #define PORT_CAP_SFP_1G (PORT_CAP_AUTONEG | PORT_CAP_100M_FDX | PORT_CAP_1G_FDX | PORT_CAP_FLOW_CTRL | PORT_CAP_SFP_ONLY)
- #define PORT_CAP_SFP_2_5G (PORT_CAP_SFP_1G | PORT_CAP_2_5G_FDX)
- #define PORT_CAP_SFP_SD_HIGH (PORT_CAP_SD_ENABLE | PORT_CAP_SD_HIGH | PORT_CAP_SD_INTERNAL | PORT_CAP_SFP_DETECT | PORT_CAP_SFP_ONLY)
- #define PORT_CAP_2_5G_TRI_SPEED_FDX (PORT_CAP_AUTONEG | PORT_CAP_2_5G_FDX | PORT_CAP_1G_FDX | PORT_CAP_100M_FDX | PORT_CAP_FLOW_CTRL)
- #define PORT_CAP_2_5G_TRI_SPEED (PORT_CAP_2_5G_TRI_SPEED_FDX | PORT_CAP_100M_HDX)
- #define PORT_CAP_2_5G_TRI_SPEED_COPPER (PORT_CAP_2_5G_TRI_SPEED | PORT_CAP_COPPER)

Typedefs

- typedef u32 port_cap_t
- typedef u64 vtss_port_counter_t

Counter type.

Enumerations

- enum vtss_port_speed_t {
 VTSS_SPEED_UNDEFINED, VTSS_SPEED_10M, VTSS_SPEED_100M, VTSS_SPEED_1G,
 VTSS_SPEED_2500M, VTSS_SPEED_5G, VTSS_SPEED_10G, VTSS_SPEED_12G }

Port speed.
- enum vtss_fiber_port_speed_t {
 VTSS_SPEED_FIBER_NOT_SUPPORTED_OR_DISABLED = 0, VTSS_SPEED_FIBER_100FX = 2,
 VTSS_SPEED_FIBER_1000X = 3, VTSS_SPEED_FIBER_AUTO = 4,
 VTSS_SPEED_FIBER_DISABLED = 5 }

Fiber Port speed.

8.4.1 Detailed Description

Port Public API Header.

This header file describes public port data-types

8.4.2 Macro Definition Documentation

8.4.2.1 PORT_CAP_NONE

```
#define PORT_CAP_NONE 0x00000000
```

No capabilities

Definition at line 46 of file port.h.

8.4.2.2 PORT_CAP_AUTONEG

```
#define PORT_CAP_AUTONEG 0x00000001
```

Auto negotiation

Definition at line 47 of file port.h.

8.4.2.3 PORT_CAP_10M_HDX

```
#define PORT_CAP_10M_HDX 0x00000002
```

10 Mbps, half duplex

Definition at line 48 of file port.h.

8.4.2.4 PORT_CAP_10M_FDX

```
#define PORT_CAP_10M_FDX 0x00000004
```

10 Mbps, full duplex

Definition at line 49 of file port.h.

8.4.2.5 PORT_CAP_100M_HDX

```
#define PORT_CAP_100M_HDX 0x00000008
```

100 Mbps, half duplex

Definition at line 50 of file port.h.

8.4.2.6 PORT_CAP_100M_FDX

```
#define PORT_CAP_100M_FDX 0x00000010
```

100 Mbps, full duplex

Definition at line 51 of file port.h.

8.4.2.7 PORT_CAP_1G_FDX

```
#define PORT_CAP_1G_FDX 0x00000020
```

1 Gbps, full duplex

Definition at line 52 of file port.h.

8.4.2.8 PORT_CAP_2_5G_FDX

```
#define PORT_CAP_2_5G_FDX 0x00000040
```

2.5 Gbps, full duplex

Definition at line 53 of file port.h.

8.4.2.9 PORT_CAP_5G_FDX

```
#define PORT_CAP_5G_FDX 0x00000080
```

5Gbps, full duplex

Definition at line 54 of file port.h.

8.4.2.10 PORT_CAP_10G_FDX

```
#define PORT_CAP_10G_FDX 0x00000100
```

10Gbps, full duplex

Definition at line 55 of file port.h.

8.4.2.11 PORT_CAP_FLOW_CTRL

```
#define PORT_CAP_FLOW_CTRL 0x00001000
```

Flow control

Definition at line 56 of file port.h.

8.4.2.12 PORT_CAP_COPPER

```
#define PORT_CAP_COPPER 0x00002000
```

Copper media

Definition at line 57 of file port.h.

8.4.2.13 PORT_CAP_FIBER

```
#define PORT_CAP_FIBER 0x00004000
```

Fiber media

Definition at line 58 of file port.h.

8.4.2.14 PORT_CAP_DUAL_COPPER

```
#define PORT_CAP_DUAL_COPPER 0x00008000
```

Dual media, copper preferred

Definition at line 59 of file port.h.

8.4.2.15 PORT_CAP_DUAL_FIBER

```
#define PORT_CAP_DUAL_FIBER 0x00010000
```

Dual media, fiber preferred

Definition at line 60 of file port.h.

8.4.2.16 PORT_CAP_SD_ENABLE

```
#define PORT_CAP_SD_ENABLE 0x00020000
```

Signal Detect enabled

Definition at line 61 of file port.h.

8.4.2.17 PORT_CAP_SD_HIGH

```
#define PORT_CAP_SD_HIGH 0x00040000
```

Signal Detect active high

Definition at line 62 of file port.h.

8.4.2.18 PORT_CAP_SD_INTERNAL

```
#define PORT_CAP_SD_INTERNAL 0x00080000
```

Signal Detect select internal

Definition at line 63 of file port.h.

8.4.2.19 PORT_CAP_DUAL_FIBER_100FX

```
#define PORT_CAP_DUAL_FIBER_100FX 0x00100000
```

Dual media (Fiber = 100FX), fiber preferred

Definition at line 64 of file port.h.

8.4.2.20 PORT_CAP_XAUI_LANE_FLIP

```
#define PORT_CAP_XAUI_LANE_FLIP 0x00200000
```

Flip the XAUI lanes

Definition at line 65 of file port.h.

8.4.2.21 PORT_CAP_VTSS_10G_PHY

```
#define PORT_CAP_VTSS_10G_PHY 0x00400000
```

Connected to VTSS 10G PHY

Definition at line 66 of file port.h.

8.4.2.22 PORT_CAP_SFP_DETECT

```
#define PORT_CAP_SFP_DETECT 0x00800000
```

Auto detect the SFP module

Definition at line 67 of file port.h.

8.4.2.23 PORT_CAP_STACKING

```
#define PORT_CAP_STACKING 0x01000000
```

Stack port candidate

Definition at line 68 of file port.h.

8.4.2.24 PORT_CAP_DUAL_SFP_DETECT

```
#define PORT_CAP_DUAL_SFP_DETECT 0x02000000
```

Auto detect the SFP module for dual media

Definition at line 69 of file port.h.

8.4.2.25 PORT_CAP_SFP_ONLY

```
#define PORT_CAP_SFP_ONLY 0x04000000
```

SFP only port (not dual media)

Definition at line 70 of file port.h.

8.4.2.26 PORT_CAP_DUAL_COPPER_100FX

```
#define PORT_CAP_DUAL_COPPER_100FX 0x08000000
```

Dual media (Fiber = 100FX), copper preferred

Definition at line 71 of file port.h.

8.4.2.27 PORT_CAP_HDX

```
#define PORT_CAP_HDX (PORT_CAP_10M_HDX | PORT_CAP_100M_HDX)
```

Half duplex

Definition at line 74 of file port.h.

8.4.2.28 PORT_CAP_TRI_SPEED_FDX

```
#define PORT_CAP_TRI_SPEED_FDX (PORT_CAP_AUTONEG | PORT_CAP_1G_FDX | PORT_CAP_100M_FDX | PORT_CAP_10M_FDX  
| PORT_CAP_FLOW_CTRL)
```

Tri-speed port full duplex only

Definition at line 75 of file port.h.

8.4.2.29 PORT_CAP_TRI_SPEED

```
#define PORT_CAP_TRI_SPEED (PORT_CAP_TRI_SPEED_FDX | PORT_CAP_HDX)
```

Tri-speed port, both full and half duplex

Definition at line 76 of file port.h.

8.4.2.30 PORT_CAP_1G_PHY

```
#define PORT_CAP_1G_PHY (PORT_CAP_COPPER | PORT_CAP_FIBER | PORT_CAP_DUAL_COPPER | PORT_CAP_DUAL_FIBER  
| PORT_CAP_DUAL_FIBER_100FX)
```

1G PHY present

Definition at line 77 of file port.h.

8.4.2.31 PORT_CAP_TRI_SPEED_COPPER

```
#define PORT_CAP_TRI_SPEED_COPPER (PORT_CAP_TRI_SPEED | PORT_CAP_COPPER)
```

Tri-speed port copper only

Definition at line 79 of file port.h.

8.4.2.32 PORT_CAP_TRI_SPEED_FIBER

```
#define PORT_CAP_TRI_SPEED_FIBER (PORT_CAP_TRI_SPEED | PORT_CAP_FIBER)
```

Tri-speed port fiber only

Definition at line 80 of file port.h.

8.4.2.33 PORT_CAP_TRI_SPEED_DUAL_COPPER

```
#define PORT_CAP_TRI_SPEED_DUAL_COPPER (PORT_CAP_TRI_SPEED | PORT_CAP_DUAL_COPPER)
```

Tri-speed port both fiber and copper. Copper prefered

Definition at line 81 of file port.h.

8.4.2.34 PORT_CAP_TRI_SPEED_DUAL_FIBER

```
#define PORT_CAP_TRI_SPEED_DUAL_FIBER (PORT_CAP_TRI_SPEED | PORT_CAP_DUAL_FIBER)
```

Tri-speed port both fiber and copper. Fiber prefered

Definition at line 82 of file port.h.

8.4.2.35 PORT_CAP_TRI_SPEED_DUAL_FIBER_100FX

```
#define PORT_CAP_TRI_SPEED_DUAL_FIBER_100FX (PORT_CAP_TRI_SPEED | PORT_CAP_DUAL_FIBER_100FX)
```

Tri-speed port both fiber (100FX) and copper. Copper prefered

Definition at line 83 of file port.h.

8.4.2.36 PORT_CAP_ANY_FIBER

```
#define PORT_CAP_ANY_FIBER (PORT_CAP_FIBER | PORT_CAP_DUAL_FIBER_100FX | PORT_CAP_DUAL_FIBER |
| PORT_CAP_DUAL_COPPER | PORT_CAP_SFP_DETECT)
```

Any fiber mode

Definition at line 84 of file port.h.

8.4.2.37 PORT_CAP_SPEED_DUAL_ANY_FIBER_FIXED_SPEED

```
#define PORT_CAP_SPEED_DUAL_ANY_FIBER_FIXED_SPEED (PORT_CAP_DUAL_FIBER_100FX | PORT_CAP_DUAL_FIBER |
| PORT_CAP_DUAL_COPPER)
```

Any fiber mode, but auto detection not supported

Definition at line 85 of file port.h.

8.4.2.38 PORT_CAP_SPEED_DUAL_ANY_FIBER

```
#define PORT_CAP_SPEED_DUAL_ANY_FIBER (PORT_CAP_DUAL_COPPER | PORT_CAP_DUAL_FIBER | PORT_CAP_DUAL_FIBER_100FX
| PORT_CAP_DUAL_SFP_DETECT)
```

Any fiber mode, auto detection supported

Definition at line 86 of file port.h.

8.4.2.39 PORT_CAP_TRI_SPEED_DUAL_ANY_FIBER

```
#define PORT_CAP_TRI_SPEED_DUAL_ANY_FIBER (PORT_CAP_TRI_SPEED | PORT_CAP_SPEED_DUAL_ANY_FIBER)
```

Copper & Fiber mode, auto detection supported

Definition at line 87 of file port.h.

8.4.2.40 PORT_CAP_TRI_SPEED_DUAL_ANY_FIBER_FIXED_SFP_SPEED

```
#define PORT_CAP_TRI_SPEED_DUAL_ANY_FIBER_FIXED_SFP_SPEED (PORT_CAP_TRI_SPEED | PORT_CAP_SPEED_DUAL_ANY_FIBER |
```

Copper & Fiber mode, but SFP auto detection not supported

Definition at line 88 of file port.h.

8.4.2.41 PORT_CAP_DUAL_FIBER_1000X

```
#define PORT_CAP_DUAL_FIBER_1000X (PORT_CAP_DUAL_FIBER | PORT_CAP_DUAL_COPPER)
```

1000Base-X fiber mode

Definition at line 89 of file port.h.

8.4.2.42 PORT_CAP_SFP_1G

```
#define PORT_CAP_SFP_1G (PORT_CAP_AUTONEG | PORT_CAP_100M_FDX | PORT_CAP_1G_FDX | PORT_CAP_FLOW_CTRL  
| PORT_CAP_SFP_ONLY)
```

SFP fiber port 100FX/1G with auto negotiation and flow control

Definition at line 91 of file port.h.

8.4.2.43 PORT_CAP_SFP_2_5G

```
#define PORT_CAP_SFP_2_5G (PORT_CAP_SFP_1G | PORT_CAP_2_5G_FDX)
```

SFP fiber port 100FX/1G/2.5G with auto negotiation and flow control

Definition at line 92 of file port.h.

8.4.2.44 PORT_CAP_SFP_SD_HIGH

```
#define PORT_CAP_SFP_SD_HIGH (PORT_CAP_SD_ENABLE | PORT_CAP_SD_HIGH | PORT_CAP_SD_INTERNAL |  
PORT_CAP_SFP_DETECT | PORT_CAP_SFP_ONLY)
```

SFP fiber port 100FX/1G/2.5G with auto negotiation and flow control, signal detect high

Definition at line 93 of file port.h.

8.4.2.45 PORT_CAP_2_5G_TRI_SPEED_FDX

```
#define PORT_CAP_2_5G_TRI_SPEED_FDX (PORT_CAP_AUTONEG | PORT_CAP_2_5G_FDX | PORT_CAP_1G_FDX |  
PORT_CAP_100M_FDX | PORT_CAP_FLOW_CTRL)
```

100M/1G/2.5G Tri-speed port full duplex only

Definition at line 95 of file port.h.

8.4.2.46 PORT_CAP_2_5G_TRI_SPEED

```
#define PORT_CAP_2_5G_TRI_SPEED (PORT_CAP_2_5G_TRI_SPEED_FDX | PORT_CAP_100M_HDX)
```

100M/1G/2.5G Tri-speed port, all full duplex and 100M half duplex

Definition at line 96 of file port.h.

8.4.2.47 PORT_CAP_2_5G_TRI_SPEED_COPPER

```
#define PORT_CAP_2_5G_TRI_SPEED_COPPER (PORT_CAP_2_5G_TRI_SPEED | PORT_CAP_COPPER)
```

100M/1G/2.5G Tri-speed port copper only

Definition at line 97 of file port.h.

8.4.3 Typedef Documentation

8.4.3.1 port_cap_t

```
typedef u32 port_cap_t
```

Bit-mask containing the port capabilities

Definition at line 99 of file port.h.

8.4.4 Enumeration Type Documentation

8.4.4.1 vtss_port_speed_t

```
enum vtss_port_speed_t
```

Port speed.

Enumerator

VTSS_SPEED_UNDEFINED	Undefined
VTSS_SPEED_10M	10 M
VTSS_SPEED_100M	100 M
VTSS_SPEED_1G	1 G
VTSS_SPEED_2500M	2.5G
VTSS_SPEED_5G	5G or 2x2.5G
VTSS_SPEED_10G	10 G
VTSS_SPEED_12G	12G

Definition at line 242 of file port.h.

8.4.4.2 vtss_fiber_port_speed_t

enum [vtss_fiber_port_speed_t](#)

Fiber Port speed.

Enumerator

<code>VTSS_SPEED_FIBER_NOT_SUPPORTED_OR_DISABLED</code>	Fiber not supported/ Fiber port disabled
<code>VTSS_SPEED_FIBER_100FX</code>	100BASE-FX
<code>VTSS_SPEED_FIBER_1000X</code>	1000BASE-X
<code>VTSS_SPEED_FIBER_AUTO</code>	Auto detection
<code>VTSS_SPEED_FIBER_DISABLED</code>	Obsolete - use <code>VTSS_SPEED_FIBER_NOT_SUPPORTED_OR_DISABLED</code> instead

Definition at line 255 of file port.h.

8.5 vtss_api/include/vtss/api/types.h File Reference

Generic types API.

```
#include <vtss/api/options.h>
#include <sys/bsdtypes.h>
```

Data Structures

- struct [vtss_aneg_t](#)
Auto negotiation struct.
- struct [vtss_vlan_tag_t](#)
- struct [vtss_mac_t](#)
MAC Address.
- struct [vtss_vid_mac_t](#)
MAC Address in specific VLAN.
- struct [vtss_packet_rx_port_conf_t](#)
Packet registration per port.
- struct [vtss_ipv6_t](#)
IPv6 address/mask.
- struct [vtss_ip_addr_t](#)
Either an IPv4 or IPv6 address.
- struct [vtss_ipv4_network_t](#)
IPv4 network.

- struct [vtss_ipv6_network_t](#)
IPv6 network.
- struct [vtss_ip_network_t](#)
IPv6 network.
- struct [vtss_ipv4_uc_t](#)
IPv4 unicast routing entry.
- struct [vtss_ipv6_uc_t](#)
IPv6 routing entry.
- struct [vtss_routing_entry_t](#)
Routing entry.
- struct [vtss_l3_counters_t](#)
Routing interface statics counter.
- struct [vtss_vcap_u8_t](#)
VCAP 8 bit value and mask.
- struct [vtss_vcap_u16_t](#)
VCAP 16 bit value and mask.
- struct [vtss_vcap_u24_t](#)
VCAP 24 bit value and mask.
- struct [vtss_vcap_u32_t](#)
VCAP 32 bit value and mask.
- struct [vtss_vcap_u40_t](#)
VCAP 40 bit value and mask.
- struct [vtss_vcap_u48_t](#)
VCAP 48 bit value and mask.
- struct [vtss_vcap_u128_t](#)
VCAP 128 bit value and mask.
- struct [vtss_vcap_vid_t](#)
VCAP VLAN ID value and mask.
- struct [vtss_vcap_ip_t](#)
VCAP IPv4 address value and mask.
- struct [vtss_vcap_udp_tcp_t](#)
VCAP UDP/TCP port range.
- struct [vtss_vcap_vr_t](#)
VCAP universal value or range.
- struct [vtss_counter_pair_t](#)
Counter pair.
- struct [vtss_evc_counters_t](#)
EVC/ECE counters.
- struct [vtss_timestamp_t](#)
Time stamp in seconds and nanoseconds.

Macros

- #define **PRIu64** "llu"
- #define **PRId64** "lli"
- #define **PRIx64** "llx"
- #define **VTSS_BIT64**(x) (1ULL << (x))
- #define **VTSS_BITMASK64**(x) ((1ULL << (x)) - 1)
- #define **VTSS_EXTRACT_BITFIELD64**(x, o, w) (((x) >> (o)) & **VTSS_BITMASK64**(w))
- #define **VTSS_ENCODE_BITFIELD64**(x, o, w) (((u64)(x) & **VTSS_BITMASK64**(w)) << (o))

- #define VTSS_ENCODE_BITMASK64(o, w) (`VTSS_BITMASK64(w) << (o)`)
- #define TRUE 1
- #define FALSE 0
- #define VTSS_PACKET_RATE_DISABLED 0xffffffff
- #define VTSS_PORT_COUNT 1
- #define VTSS_PORT_COUNT 31
- #define VTSS_PORTS VTSS_OPT_PORT_COUNT
- #define VTSS_PORT_NO_NONE (0xffffffff)
- #define VTSS_PORT_NO_CPU (0xfffffff)
- #define VTSS_PORT_NO_START (0)
- #define VTSS_PORT_NO_END (VTSS_PORT_NO_START+VTSS_PORTS)
- #define VTSS_PORT_ARRAY_SIZE VTSS_PORT_NO_END
- #define VTSS_PORT_IS_PORT(x) ((x)<VTSS_PORT_NO_END)
- #define VTSS_PRIOS 8
- #define VTSS_PRIO_NO_NONE 0xffffffff
- #define VTSS_PRIO_START 0
- #define VTSS_PRIO_END (VTSS_PRIO_START + VTSS_PRIOS)
- #define VTSS_PRIO_ARRAY_SIZE VTSS_PRIO_END
- #define VTSS_QUEUES VTSS_PRIOS
- #define VTSS_QUEUE_START 0
- #define VTSS_QUEUE_END (VTSS_QUEUE_START + VTSS_QUEUES)
- #define VTSS_QUEUE_ARRAY_SIZE VTSS_QUEUE_END
- #define VTSS_PCPS 8
- #define VTSS_PCP_START 0
- #define VTSS_PCP_END (VTSS_PCP_START + VTSS_PCPS)
- #define VTSS_PCP_ARRAY_SIZE VTSS_PCP_END
- #define VTSS_DEIS 2
- #define VTSS_DEI_START 0
- #define VTSS_DEI_END (VTSS_DEI_START + VTSS_DEIS)
- #define VTSS_DEI_ARRAY_SIZE VTSS_DEI_END
- #define VTSS_DPLS 2
- #define VTSS_DPLS 4
- #define VTSS_DPL_START 0
- #define VTSS_DPL_END (VTSS_DPL_START + VTSS_DPLS)
- #define VTSS_DPL_ARRAY_SIZE VTSS_DPL_END
- #define VTSS_BITRATE_DISABLED 0xffffffff
- #define VTSS_VID_NULL ((const `vtss_vid_t`)0)
- #define VTSS_VID_DEFAULT ((const `vtss_vid_t`)1)
- #define VTSS_VID_RESERVED ((const `vtss_vid_t`)0FFF)
- #define VTSS_VIDS ((const `vtss_vid_t`)4096)
- #define VTSS_VID_ALL ((const `vtss_vid_t`)0x1000)
- #define VTSSETYPE_VTSS 0x8880
- #define VTSS_MAC_ADDR_SZ_BYTES 6
- #define MAC_ADDR_BROADCAST {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF}
- #define VTSS_EVCS 4096
- #define VTSS_ISDX_NONE (0)
- #define VTSS_AGGRS (VTSS_PORTS/2)
- #define VTSS_AGGR_NO_NONE 0xffffffff
- #define VTSS_AGGR_NO_START 0
- #define VTSS_AGGR_NO_END (VTSS_AGGR_NO_START+VTSS_AGGRS)
- #define VTSS_GLAGS 32
- #define VTSS_GLAG_NO_NONE 0xffffffff
- #define VTSS_GLAG_NO_START 0
- #define VTSS_GLAG_NO_END (VTSS_GLAG_NO_START+VTSS_GLAGS)
- #define VTSS_GLAG_PORTS 8

- #define VTSS_GLAG_PORT_START 0
- #define VTSS_GLAG_PORT_END (VTSS_GLAG_PORT_START+VTSS_GLAG_PORTS)
- #define VTSS_GLAG_PORT_ARRAY_SIZE VTSS_GLAG_PORT_END
- #define VTSS_PACKET_RX_QUEUE_CNT 10
- #define VTSS_PACKET_RX_GRP_CNT 4
- #define VTSS_PACKET_TX_GRP_CNT 5
- #define VTSS_PACKET_RX_QUEUE_NONE (0xffffffff)
- #define VTSS_PACKET_RX_QUEUE_START (0)
- #define VTSS_PACKET_RX_QUEUE_END (VTSS_PACKET_RX_QUEUE_START + VTSS_PACKET_RX_QUEUE_CNT)
- #define VTSS_ACL_POLICERS 16
- #define VTSS_ACL_POLICER_NO_START 0
- #define VTSS_ACL_POLICER_NO_END (VTSS_ACL_POLICER_NO_START + VTSS_ACL_POLICERS)
- #define VTSS_ACL_POLICY_NO_NONE 0xffffffff
- #define VTSS_ACL_POLICY_NO_MIN 0
- #define VTSS_ACL_POLICY_NO_MAX 255
- #define VTSS_ACL_POLICIES (VTSS_ACL_POLICY_NO_MAX + 1)
- #define VTSS_ACL_POLICY_NO_START VTSS_ACL_POLICY_NO_MIN
- #define VTSS_ACL_POLICY_NO_END (VTSS_ACL_POLICY_NO_START + VTSS_ACL_POLICIES)
- #define VTSS_HQOS_COUNT 256
- #define VTSS_HQOS_ID_NONE 0xffff
- #define VTSS_ONE_MIA 1000000000
- #define VTSS_ONE_MILL 1000000
- #define VTSS_MAX_TIMEINTERVAL 0x7fffffffffffffLL
- #define VTSS_INTERVAL_SEC(t) ((i32)VTSS_DIV64((t)>>16, VTSS_ONE_MIA))
- #define VTSS_INTERVAL_MS(t) ((i32)VTSS_DIV64((t)>>16, VTSS_ONE_MILL))
- #define VTSS_INTERVAL_US(t) ((i32)VTSS_DIV64((t)>>16, 1000))
- #define VTSS_INTERVAL_NS(t) ((i32)VTSS_MOD64((t)>>16, VTSS_ONE_MIA))
- #define VTSS_INTERVAL_PS(t) (((((i32)(t & 0xffff))*1000)+0x8000)/0x10000)
- #define VTSS_SEC_NS_INTERVAL(s, n) (((vtss_timeinterval_t)(n)+(vtss_timeinterval_t)(s)*VTSS_ONE_MIA)<<16)
- #define VTSS_CLOCK_IDENTITY_LENGTH 8
- #define VTSS_SYNC_CLK_PORT_ARRAY_SIZE 2

Typedefs

- typedef char i8

Fallback Integer types.
- typedef signed short i16
- typedef signed int i32
- typedef signed long long i64
- typedef unsigned char u8
- typedef unsigned short u16
- typedef unsigned int u32
- typedef unsigned long long u64
- typedef unsigned char BOOL
- typedef unsigned int uintptr_t
- typedef int vtss_rc

Error code type.
- typedef u32 vtss_chip_no_t

Chip number used for targets with multiple chips.
- typedef struct vtss_state_s * vtss_inst_t

Instance identifier.
- typedef BOOL vtss_event_t

Description: Event type. When a variable of this type is used as an input parameter, the API will set the variable if the event has occurred. The API will never clear the variable. It is up to the application to clear the variable, when the event has been handled.

- **typedef u32 vtss_packet_rate_t**
Policer packet rate in PPS.
- **typedef u32 vtss_port_no_t**
Port Number.
- **typedef u32 vtss_phys_port_no_t**
Physical port number.
- **typedef u32 vtss_prio_t**
Priority number.
- **typedef u32 vtss_queue_t**
Queue number.
- **typedef u32 vtss_tagprio_t**
Tag Priority or Priority Code Point (PCP)
- **typedef BOOL vtss_dei_t**
Drop Eligible Indicator (DEI)
- **typedef u8 vtss_dp_level_t**
Drop Precedence Level (DPL)
- **typedef u8 vtss_pct_t**
Percentage, 0-100.
- **typedef u32 vtss_bitrate_t**
Policer/Shaper bit rate in kbps (1000 bits per second). The rate will be rounded to the nearest value supported by the chip.
- **typedef u32 vtss_burst_level_t**
Policer/shaper burst level in bytes. The level will be rounded to the nearest value supported by the chip.
- **typedef u8 vtss_dscp_t**
DSCP value (0-63)
- **typedef u32 vtss_qce_id_t**
QoS Control Entry ID.
- **typedef u16 vtss_evc_policer_id_t**
EVC policer index.
- **typedef u32 vtss_wred_group_t**
WRED group number.
- **typedef u16 vtss_vid_t**
VLAN Identifier.
- **typedef u16 vtss_etype_t**
Ethernet Type.
- **typedef u8 vtss_mac_addr_t[VTSS_MAC_ADDR_SZ_BYTES]**
- **typedef u16 vtss_evc_id_t**
EVC ID.
- **typedef u32 vtss_isdx_t**
- **typedef u32 vtss_aggr_no_t**
Aggregation Number.
- **typedef u32 vtss_glag_no_t**
Description: GLAG number.
- **typedef u32 vtss_packet_rx_queue_t**
Description: CPU Rx queue number.
- **typedef u32 vtss_packet_rx_grp_t**
Description: CPU Rx group number.
- **typedef u32 vtss_packet_tx_grp_t**

- **typedef u16 vtss_udp_tcp_t**
Description: CPU Tx group number.
- **typedef u32 vtss_ip_t**
Description: UDP/TCP port number.
- **typedef vtss_ip_t vtss_ipv4_t**
IPv4 address/mask.
- **typedef u32 vtss_prefix_size_t**
IPv4 address/mask.
- **typedef u16 vtss_vcap_vr_value_t**
VCAP universal value or range type.
- **typedef u32 vtss_acl_policer_no_t**
ACL policer number.
- **typedef u32 vtss_acl_policy_no_t**
ACL policy number.
- **typedef u32 vtss_ece_id_t**
EVC Control Entry (ECE) ID.
- **typedef u64 vtss_counter_t**
Counter.
- **typedef u16 vtss_hqos_id_t**
HQoS entry identifier (HQoS ID)
- **typedef i64 vtss_clk_adj_rate_t**
*Clock adjustment rate in parts per billion (ppb) * 1<<16. Range is +2**47 ppb For example, 8.25 ppb is expressed as 0x0000.0000.0008.4000.*
- **typedef i64 vtss_timeinterval_t**
*Time interval in ns * 1<<16 range +2**47 ns = 140737 sec = 39 hours For example, 2.5 ns is expressed as 0x0000.0000.0002.8000.*
- **typedef u8 vtss_clock_identity[VTSS_CLOCK_IDENTITY_LENGTH]**
PTP clock unique identifier.

Enumerations

- **enum {**
- VTSS_RC_OK** = 0, **VTSS_RC_ERROR** = -1, **VTSS_RC_INV_STATE** = -2, **VTSS_RC_INCOMPLETE** = -3, **VTSS_RC_ERR_CLK_CONF_NOT_SUPPORTED** = -6, **VTSS_RC_ERR_KR_CONF_NOT_SUPPORTED** = -7, **VTSS_RC_ERR_KR_CONF_INVALID_PARAMETER** = -8, **VTSS_RC_ERR_PHY_BASE_NO_NOT_FOUND** = -50, **VTSS_RC_ERR_PHY_6G_MACRO_SETUP** = -51, **VTSS_RC_ERR_PHY_MEDIA_IF_NOT_SUPPORTED** = -52, **VTSS_RC_ERR_PHY_CLK_CONF_NOT_SUPPORTED** = -53, **VTSS_RC_ERR_PHY_GPIO_ALT_MODE_NOT_SUPPORTED** = -54, **VTSS_RC_ERR_PHY_GPIO_PIN_NOT_SUPPORTED** = -55, **VTSS_RC_ERR_PHY_PORT_OUT_RANGE** = -56, **VTSS_RC_ERR_PHY_PATCH_SETTING_NOT_SUPPORTED** = -57, **VTSS_RC_ERR_PHY_LCPLL_NOT_SUPPORTED** = -58, **VTSS_RC_ERR_PHY_RCPLL_NOT_SUPPORTED** = -59, **VTSS_RC_ERR_MACSEC_INVALID_SCI_MACADDR** = -60, **VTSS_RC_ERR_MACSEC_NOT_ENABLED** = -61, **VTSS_RC_ERR_MACSEC_SECY_ALREADY_IN_USE** = -63, **VTSS_RC_ERR_MACSEC_NO_SECY_FOUND** = -64, **VTSS_RC_ERR_MACSEC_NO_SECY_VACANCY** = -65, **VTSS_RC_ERR_MACSEC_INVALID_VALIDATE_FRM** = -66, **VTSS_RC_ERR_MACSEC_COULD_NOT_PRG_SA_MAC** = -67, **VTSS_RC_ERR_MACSEC_COULD_NOT_PRG_SA_FLOW** = -68, **VTSS_RC_ERR_MACSEC_COULD_NOT_ENA_SA** = -69, **VTSS_RC_ERR_MACSEC_COULD_NOT_SET_SA** = -70, **VTSS_RC_ERR_MACSEC_INVALID_BYPASS_HDR_LEN** = -71, **VTSS_RC_ERR_MACSEC_SC_NOT_FOUND** = -72, **VTSS_RC_ERR_MACSEC_NO_CTRL_FRM_MATCH**

```
= -73, VTSS_RC_ERR_MACSEC_COULD_NOT_SET_PATTERN = -74, VTSS_RC_ERR_MACSEC_TIMEOUT_ISSUE
= -75,
VTSS_RC_ERR_MACSEC_COULD_NOT_EMPTY_EGRESS = -76, VTSS_RC_ERR_MACSEC_AN_NOT_CREATED
= -77, VTSS_RC_ERR_MACSEC_COULD_NOT_EMPTY_INGRESS = -78, VTSS_RC_ERR_MACSEC_TX_SC_NOT_EXIST
= -80,
VTSS_RC_ERR_MACSEC_COULD_NOT_DISABLE_SA = -81, VTSS_RC_ERR_MACSEC_COULD_NOT_DEL_RX_SA
= -82, VTSS_RC_ERR_MACSEC_COULD_NOT_DEL_TX_SA = -83, VTSS_RC_ERR_MACSEC_PATTERN_NOT_SET
= -84,
VTSS_RC_ERR_MACSEC_HW_RESOURCE_EXHUSTED = -85, VTSS_RC_ERR_MACSEC_SCI_ALREADY_EXISTS
= -86, VTSS_RC_ERR_MACSEC_SC_RESOURCE_NOT_FOUND = -87, VTSS_RC_ERR_MACSEC_RX_AN_ALREADY_IN_
= -88,
VTSS_RC_ERR_MACSEC_EMPTY_RECORD = -89, VTSS_RC_ERR_MACSEC_COULD_NOT_PRG_XFORM
= -90, VTSS_RC_ERR_MACSEC_COULD_NOT_TOGGLE_SA = -91, VTSS_RC_ERR_MACSEC_TX_AN_ALREADY_IN_US
= -92,
VTSS_RC_ERR_MACSEC_ALL_AVAILABLE_SA_IN_USE = -93, VTSS_RC_ERR_MACSEC_MATCH_DISABLE
= -94, VTSS_RC_ERR_MACSEC_ALL_CP_RULES_IN_USE = -95, VTSS_RC_ERR_MACSEC_PATTERN_PRIO_NOT_VAL
= -96,
VTSS_RC_ERR_MACSEC_BUFFER_TOO_SMALL = -97, VTSS_RC_ERR_MACSEC_FRAME_TOO_LONG
= -98, VTSS_RC_ERR_MACSEC_FRAME_TRUNCATED = -99, VTSS_RC_ERR_MACSEC_PHY_POWERED_DOWN
= -100,
VTSS_RC_ERR_MACSEC_PHY_NOT_MACSEC_CAPABLE = -101, VTSS_RC_ERR_MACSEC_AN_NOT_EXIST
= -102, VTSS_RC_ERR_MACSEC_NO_PATTERN_CFG = -103, VTSS_RC_ERR_MACSEC_MAX_MTU =
-105,
VTSS_RC_ERR_MACSEC_UNEXPECT_CP_MODE = -106, VTSS_RC_ERR_MACSEC_COULD_NOT_DISABLE_AN
= -107, VTSS_RC_ERR_MACSEC_RULE_OUT_OF_RANGE = -108, VTSS_RC_ERR_MACSEC_RULE_NOT_EXIST
= -109,
VTSS_RC_ERR_MACSEC_CSR_READ = -110, VTSS_RC_ERR_MACSEC_CSR_WRITE = -111,
VTSS_RC_ERR_PHY_6G_RCPLL_ON_BASE_PORT_ONLY = -112, VTSS_RC_ERR_INVALID_NULL_PTR
= -200 }
```

Error codes.

- enum `vtss_mem_flags_t` { `VTSS_MEM_FLAGS_NONE` = 0x0, `VTSS_MEM_FLAGS_DMA` = 0x1,
`VTSS_MEM_FLAGS_PERSIST` = 0x2 }

Memory allocation flags.

- enum `vtss_port_interface_t` {
`VTSS_PORT_INTERFACE_NO_CONNECTION`, `VTSS_PORT_INTERFACE_LOOPBACK`, `VTSS_PORT_INTERFACE_INTE`,
`VTSS_PORT_INTERFACE_MII`,
`VTSS_PORT_INTERFACE_GMII`, `VTSS_PORT_INTERFACE_RGMII`, `VTSS_PORT_INTERFACE_TBI`,
`VTSS_PORT_INTERFACE_RTBI`,
`VTSS_PORT_INTERFACE_SGMII`, `VTSS_PORT_INTERFACE_SGMII_CISCO`, `VTSS_PORT_INTERFACE_SERDES`,
`VTSS_PORT_INTERFACE_VAUI`,
`VTSS_PORT_INTERFACE_100FX`, `VTSS_PORT_INTERFACE_XAUI`, `VTSS_PORT_INTERFACE_RXAUI`,
`VTSS_PORT_INTERFACE_XGMII`,
`VTSS_PORT_INTERFACE_SPI4`, `VTSS_PORT_INTERFACE_QSGMII`, `VTSS_PORT_INTERFACE_SFI` }

The different interfaces for connecting MAC and PHY.

- enum `vtss_serdes_mode_t` {
`VTSS_SERDES_MODE_DISABLE`, `VTSS_SERDES_MODE_XAUI_12G`, `VTSS_SERDES_MODE_XAUI`,
`VTSS_SERDES_MODE_RXAUI`,
`VTSS_SERDES_MODE_RXAUI_12G`, `VTSS_SERDES_MODE_2G5`, `VTSS_SERDES_MODE_QSGMII`,
`VTSS_SERDES_MODE_SGMII`,
`VTSS_SERDES_MODE_100FX`, `VTSS_SERDES_MODE_1000BaseX`, `VTSS_SERDES_MODE_SFI`,
`VTSS_SERDES_MODE_SFI_DAC`,
`VTSS_SERDES_MODE_IDLE` }

Serdes macro mode.

- enum `vtss_storm_policer_mode_t` { `VTSS_STORM_POLICER_MODE_PORTS_AND_CPU`, `VTSS_STORM_POLICER_MODE_B`,
`VTSS_STORM_POLICER_MODE_CPU_ONLY` }

Storm policer mode configuration.

- enum `vtss_policer_type_t` { `VTSS_POLICER_TYPE_MEF`, `VTSS_POLICER_TYPE_SINGLE` }

- Dual leaky buckets policer configuration.*
- enum `vtss_vlan_frame_t`{ `VTSS_VLAN_FRAME_ALL`, `VTSS_VLAN_FRAME_TAGGED`, `VTSS_VLAN_FRAME_UNTAGGED` }

VLAN acceptable frame type.

 - enum `vtss_packet_reg_type_t`{
`VTSS_PACKET_REG_NORMAL`, `VTSS_PACKET_REG_FORWARD`, `VTSS_PACKET_REG_DISCARD`,
`VTSS_PACKET_REG_CPU_COPY`,
`VTSS_PACKET_REG_CPU_ONLY` }

Packet registration type.

 - enum `vtss_vdd_t`{ `VTSS_VDD_1V0`, `VTSS_VDD_1V2` }

VDD power supply.

 - enum `vtss_ip_type_t`{ `VTSS_IP_TYPE_NONE` = 0, `VTSS_IP_TYPE_IPV4` = 1, `VTSS_IP_TYPE_IPV6` = 2 }

IP address type.

 - enum `vtss_routing_entry_type_t`{ `VTSS_ROUTING_ENTRY_TYPE_INVALID` = 0, `VTSS_ROUTING_ENTRY_TYPE_IPV6_UC` = 1, `VTSS_ROUTING_ENTRY_TYPE_IPV4_MC` = 2, `VTSS_ROUTING_ENTRY_TYPE_IPV4_UC` = 3 }

Routing entry type.

 - enum `vtss_vcap_bit_t`{ `VTSS_VCAP_BIT_ANY`, `VTSS_VCAP_BIT_0`, `VTSS_VCAP_BIT_1` }

VCAP 1 bit.

 - enum `vtss_vcap_vr_type_t`{ `VTSS_VCAP_VR_TYPE_VALUE_MASK`, `VTSS_VCAP_VR_TYPE_RANGE_INCLUSIVE`,
`VTSS_VCAP_VR_TYPE_RANGE_EXCLUSIVE` }

Value/Range type.

 - enum `vtss_vcap_key_type_t`{ `VTSS_VCAP_KEY_TYPE_NORMAL`, `VTSS_VCAP_KEY_TYPE_DOUBLE_TAG`,
`VTSS_VCAP_KEY_TYPE_IP_ADDR`, `VTSS_VCAP_KEY_TYPE_MAC_IP_ADDR` }

VCAP key type.

 - enum `vtss_ece_dir_t`{ `VTSS_ECE_DIR_BOTH`, `VTSS_ECE_DIR_UNI_TO_NNI`, `VTSS_ECE_DIR_NNI_TO_UNI` }

ECE direction.

 - enum `vtss_ece_pop_tag_t`{ `VTSS_ECE_POP_TAG_0`, `VTSS_ECE_POP_TAG_1`, `VTSS_ECE_POP_TAG_2` }

Ingress tag popping.

 - enum `vtss_ece_inner_tag_type_t` { `VTSS_ECE_INNER_TAG_NONE`, `VTSS_ECE_INNER_TAG_C`,
`VTSS_ECE_INNER_TAG_S`, `VTSS_ECE_INNER_TAG_S_CUSTOM` }

ECE inner tag type.

 - enum `vtss_hqos_sch_mode_t`{ `VTSS_HQOS_SCH_MODE_NORMAL`, `VTSS_HQOS_SCH_MODE_BASIC`,
`VTSS_HQOS_SCH_MODE_HIERARCHICAL` }

HQoS port scheduling mode.

8.5.1 Detailed Description

Generic types API.

This header file describes generic types used in the API

8.5.2 Macro Definition Documentation

8.5.2.1 PRIu64

```
#define PRIu64 "llu"
```

Fallback un-signed 64-bit formatting string

Definition at line 111 of file types.h.

8.5.2.2 PRIi64

```
#define PRIi64 "lli"
```

Fallback signed 64-bit formatting string

Definition at line 115 of file types.h.

8.5.2.3 PRIx64

```
#define PRIx64 "llx"
```

Fallback hex 64-bit formatting string

Definition at line 119 of file types.h.

8.5.2.4 VTSS_BIT64

```
#define VTSS_BIT64( x ) (1ULL << (x))
```

Set one bit in a 64-bit mask

Definition at line 122 of file types.h.

8.5.2.5 VTSS_BITMASK64

```
#define VTSS_BITMASK64( x ) ((1ULL << (x)) - 1)
```

Get a bitmask consisting of x ones

Definition at line 123 of file types.h.

8.5.2.6 VTSS_EXTRACT_BITFIELD64

```
#define VTSS_EXTRACT_BITFIELD64(  
    x,  
    o,  
    w) (((x) >> (o)) & VTSS_BITMASK64(w))
```

Extract w bits from bit position o in x

Definition at line 124 of file types.h.

8.5.2.7 VTSS_ENCODE_BITFIELD64

```
#define VTSS_ENCODE_BITFIELD64(  
    x,  
    o,  
    w) (((u64)(x) & VTSS_BITMASK64(w)) << (o))
```

Place w bits of x at bit position o

Definition at line 125 of file types.h.

8.5.2.8 VTSS_ENCODE_BITMASK64

```
#define VTSS_ENCODE_BITMASK64(  
    o,  
    w) (VTSS_BITMASK64(w) << (o))
```

Create a bitmask of w bits positioned at o

Definition at line 126 of file types.h.

8.5.2.9 TRUE

```
#define TRUE 1
```

True boolean value

Definition at line 129 of file types.h.

8.5.2.10 FALSE

```
#define FALSE 0
```

False boolean value

Definition at line 132 of file types.h.

8.5.2.11 VTSS_PACKET_RATE_DISABLED

```
#define VTSS_PACKET_RATE_DISABLED 0xffffffff
```

Special value for disabling packet policer

Definition at line 238 of file types.h.

8.5.2.12 VTSS_PORT_COUNT [1/2]

```
#define VTSS_PORT_COUNT 1
```

Default number of ports

Number of ports

Definition at line 396 of file types.h.

8.5.2.13 VTSS_PORT_COUNT [2/2]

```
#define VTSS_PORT_COUNT 31
```

Default number of ports

Number of ports

Definition at line 396 of file types.h.

8.5.2.14 VTSS_PORTS

```
#define VTSS_PORTS VTSS_OPT_PORT_COUNT
```

Number of ports

Definition at line 442 of file types.h.

8.5.2.15 VTSS_PORT_NO_NONE

```
#define VTSS_PORT_NO_NONE (0xffffffff)
```

Port number none

Definition at line 448 of file types.h.

8.5.2.16 VTSS_PORT_NO_CPU

```
#define VTSS_PORT_NO_CPU (0xfffffff)
```

Port number for CPU for special purposes

Definition at line 449 of file types.h.

8.5.2.17 VTSS_PORT_NO_START

```
#define VTSS_PORT_NO_START (0)
```

Port start number

Definition at line 450 of file types.h.

8.5.2.18 VTSS_PORT_NO_END

```
#define VTSS_PORT_NO_END (VTSS_PORT_NO_START+VTSS_PORTS)
```

Port end number

Definition at line 451 of file types.h.

8.5.2.19 VTSS_PORT_ARRAY_SIZE

```
#define VTSS_PORT_ARRAY_SIZE VTSS_PORT_NO_END
```

Port number array size

Definition at line 452 of file types.h.

8.5.2.20 VTSS_PORT_IS_PORT

```
#define VTSS_PORT_IS_PORT( x ) ((x)<VTSS_PORT_NO_END)
```

Valid port number

Definition at line 454 of file types.h.

8.5.2.21 VTSS_PRIOS

```
#define VTSS_PRIOS 8
```

Number of priorities

Definition at line 515 of file types.h.

8.5.2.22 VTSS_PRIO_NO_NONE

```
#define VTSS_PRIO_NO_NONE 0xffffffff
```

Priority number none (= undefined)

Definition at line 516 of file types.h.

8.5.2.23 VTSS_PRIO_START

```
#define VTSS_PRIO_START 0
```

Priority start number (lowest)

Definition at line 517 of file types.h.

8.5.2.24 VTSS_PRIO_END

```
#define VTSS_PRIO_END (VTSS_PRIO_START + VTSS_PRIOS)
```

Priority end number

Definition at line 518 of file types.h.

8.5.2.25 VTSS_PRIO_ARRAY_SIZE

```
#define VTSS_PRIO_ARRAY_SIZE VTSS_PRIO_END
```

Priority number array size

Definition at line 519 of file types.h.

8.5.2.26 VTSS_QUEUES

```
#define VTSS_QUEUES VTSS_PRIOS
```

Number of queues

Definition at line 523 of file types.h.

8.5.2.27 VTSS_QUEUE_START

```
#define VTSS_QUEUE_START 0
```

Queue start number

Definition at line 524 of file types.h.

8.5.2.28 VTSS_QUEUE_END

```
#define VTSS_QUEUE_END (VTSS_QUEUE_START + VTSS_QUEUES)
```

Queue end number

Definition at line 525 of file types.h.

8.5.2.29 VTSS_QUEUE_ARRAY_SIZE

```
#define VTSS_QUEUE_ARRAY_SIZE VTSS_QUEUE_END
```

Queue number array size

Definition at line 526 of file types.h.

8.5.2.30 VTSS_PCPS

```
#define VTSS_PCPS 8
```

Number of PCP values

Definition at line 530 of file types.h.

8.5.2.31 VTSS_PCP_START

```
#define VTSS_PCP_START 0
```

PCP start number

Definition at line 531 of file types.h.

8.5.2.32 VTSS_PCP_END

```
#define VTSS_PCP_END (VTSS_PCP_START + VTSS_PCPS)
```

PCP end number

Definition at line 532 of file types.h.

8.5.2.33 VTSS_PCP_ARRAY_SIZE

```
#define VTSS_PCP_ARRAY_SIZE VTSS_PCP_END
```

PCP array size

Definition at line 533 of file types.h.

8.5.2.34 VTSS_DEIS

```
#define VTSS_DEIS 2
```

Number of DEI values

Definition at line 537 of file types.h.

8.5.2.35 VTSS_DEI_START

```
#define VTSS_DEI_START 0
```

DEI start number

Definition at line 538 of file types.h.

8.5.2.36 VTSS_DEI_END

```
#define VTSS_DEI_END (VTSS_DEI_START + VTSS_DEIS)
```

DEI end number

Definition at line 539 of file types.h.

8.5.2.37 VTSS_DEI_ARRAY_SIZE

```
#define VTSS_DEI_ARRAY_SIZE VTSS_DEI_END
```

DEI array size

Definition at line 540 of file types.h.

8.5.2.38 VTSS_DPLS [1/2]

```
#define VTSS_DPLS 2
```

Default number of drop precedence levels

Number of drop precedence levels

Definition at line 548 of file types.h.

8.5.2.39 VTSS_DPLS [2/2]

```
#define VTSS_DPLS 4
```

Default number of drop precedence levels

Number of drop precedence levels

Definition at line 548 of file types.h.

8.5.2.40 VTSS_DPL_START

```
#define VTSS_DPL_START 0
```

DPL start number

Definition at line 551 of file types.h.

8.5.2.41 VTSS_DPL_END

```
#define VTSS_DPL_END (VTSS_DPL_START + VTSS_DPLS)
```

DPL end number

Definition at line 552 of file types.h.

8.5.2.42 VTSS_DPL_ARRAY_SIZE

```
#define VTSS_DPL_ARRAY_SIZE VTSS_DPL_END
```

DPL array size

Definition at line 553 of file types.h.

8.5.2.43 VTSS_BITRATE_DISABLED

```
#define VTSS_BITRATE_DISABLED 0xffffffff
```

Bitrate disabled

Definition at line 563 of file types.h.

8.5.2.44 VTSS_VID_NULL

```
#define VTSS_VID_NULL ((const vtss_vid_t)0)
```

NULL VLAN ID

Definition at line 609 of file types.h.

8.5.2.45 VTSS_VID_DEFAULT

```
#define VTSS_VID_DEFAULT ((const vtss_vid_t)1)
```

Default VLAN ID

Definition at line 610 of file types.h.

8.5.2.46 VTSS_VID_RESERVED

```
#define VTSS_VID_RESERVED ((const vtss_vid_t)0xFFFF)
```

Reserved VLAN ID

Definition at line 611 of file types.h.

8.5.2.47 VTSS_VIDS

```
#define VTSS_VIDS ((const vtss_vid_t)4096)
```

Number of VLAN IDs

Definition at line 612 of file types.h.

8.5.2.48 VTSS_VID_ALL

```
#define VTSS_VID_ALL ((const vtss_vid_t)0x1000)
```

Untagged VID: All VLAN IDs

Definition at line 613 of file types.h.

8.5.2.49 VTSS_ETYPE_VTSS

```
#define VTSS_ETYPE_VTSS 0x8880
```

Vitesse Ethernet Type

Definition at line 640 of file types.h.

8.5.2.50 VTSS_MAC_ADDR_SZ_BYTES

```
#define VTSS_MAC_ADDR_SZ_BYTES 6
```

Number of bytes for representing MAC address (SMAC/DMAC) type

Definition at line 648 of file types.h.

8.5.2.51 MAC_ADDR_BROADCAST

```
#define MAC_ADDR_BROADCAST {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF}
```

Broadcast address used for addr in the [vtss_mac_t](#) struct

Definition at line 658 of file types.h.

8.5.2.52 VTSS_EVCS

```
#define VTSS_EVCS 4096
```

Maximum number of Ethernet Virtual Connections

Definition at line 664 of file types.h.

8.5.2.53 VTSS_ISDX_NONE

```
#define VTSS_ISDX_NONE (0)
```

Ingress Service Index number none

Definition at line 674 of file types.h.

8.5.2.54 VTSS_AGGRS

```
#define VTSS_AGGRS (VTSS_PORTS/2)
```

Number of LLAGs

Definition at line 678 of file types.h.

8.5.2.55 VTSS_AGGR_NO_NONE

```
#define VTSS_AGGR_NO_NONE 0xffffffff
```

Aggregation number none

Definition at line 679 of file types.h.

8.5.2.56 VTSS_AGGR_NO_START

```
#define VTSS_AGGR_NO_START 0
```

Aggregation start number

Definition at line 680 of file types.h.

8.5.2.57 VTSS_AGGR_NO_END

```
#define VTSS_AGGR_NO_END (VTSS_AGGR_NO_START+VTSS_AGGRS)
```

Aggregation number end

Definition at line 681 of file types.h.

8.5.2.58 VTSS_GLAGS

```
#define VTSS_GLAGS 32
```

Number of GLAGs

Definition at line 687 of file types.h.

8.5.2.59 VTSS_GLAG_NO_NONE

```
#define VTSS_GLAG_NO_NONE 0xffffffff
```

GLAG number none

Definition at line 691 of file types.h.

8.5.2.60 VTSS_GLAG_NO_START

```
#define VTSS_GLAG_NO_START 0
```

GLAG start number

Definition at line 692 of file types.h.

8.5.2.61 VTSS_GLAG_NO_END

```
#define VTSS_GLAG_NO_END (VTSS_GLAG_NO_START+VTSS_GLAGS)
```

GLAG end number

Definition at line 693 of file types.h.

8.5.2.62 VTSS_GLAG_PORTS

```
#define VTSS_GLAG_PORTS 8
```

Number of GLAG ports

Definition at line 696 of file types.h.

8.5.2.63 VTSS_GLAG_PORT_START

```
#define VTSS_GLAG_PORT_START 0
```

GLAG port start number

Definition at line 697 of file types.h.

8.5.2.64 VTSS_GLAG_PORT_END

```
#define VTSS_GLAG_PORT_END (VTSS_GLAG_PORT_START+VTSS_GLAG_PORTS)
```

GLAG port end number

Definition at line 698 of file types.h.

8.5.2.65 VTSS_GLAG_PORT_ARRAY_SIZE

```
#define VTSS_GLAG_PORT_ARRAY_SIZE VTSS_GLAG_PORT_END
```

GLAG port array size

Definition at line 699 of file types.h.

8.5.2.66 VTSS_PACKET_RX_QUEUE_CNT

```
#define VTSS_PACKET_RX_QUEUE_CNT 10
```

Number of Rx packet queues. The last two are only usable as super priority queues.

Definition at line 738 of file types.h.

8.5.2.67 VTSS_PACKET_RX_GRP_CNT

```
#define VTSS_PACKET_RX_GRP_CNT 4
```

Number of Rx packet groups to which any queue can map

Definition at line 740 of file types.h.

8.5.2.68 VTSS_PACKET_TX_GRP_CNT

```
#define VTSS_PACKET_TX_GRP_CNT 5
```

Number of Tx packet groups

Definition at line 742 of file types.h.

8.5.2.69 VTSS_PACKET_RX_QUEUE_NONE

```
#define VTSS_PACKET_RX_QUEUE_NONE (0xffffffff)
```

Rx queue not selected for a particular type of frames

Definition at line 745 of file types.h.

8.5.2.70 VTSS_PACKET_RX_QUEUE_START

```
#define VTSS_PACKET_RX_QUEUE_START (0)
```

Rx queue start number

Definition at line 746 of file types.h.

8.5.2.71 VTSS_PACKET_RX_QUEUE_END

```
#define VTSS_PACKET_RX_QUEUE_END (VTSS_PACKET_RX_QUEUE_START + VTSS_PACKET_RX_QUEUE_CNT)
```

Rx queue end number

Definition at line 747 of file types.h.

8.5.2.72 VTSS_ACL_POLICERS

```
#define VTSS_ACL_POLICERS 16
```

Number of ACL policers

Definition at line 1028 of file types.h.

8.5.2.73 VTSS_ACL_POLICER_NO_START

```
#define VTSS_ACL_POLICER_NO_START 0
```

ACL policer start number

Definition at line 1029 of file types.h.

8.5.2.74 VTSS_ACL_POLICER_NO_END

```
#define VTSS_ACL_POLICER_NO_END (VTSS_ACL_POLICER_NO_START + VTSS_ACL_POLICERS)
```

ACL policer end number

Definition at line 1030 of file types.h.

8.5.2.75 VTSS_ACL_POLICY_NO_NONE

```
#define VTSS_ACL_POLICY_NO_NONE 0xffffffff
```

ACLs disabled on port

Definition at line 1034 of file types.h.

8.5.2.76 VTSS_ACL_POLICY_NO_MIN

```
#define VTSS_ACL_POLICY_NO_MIN 0
```

ACLs policy minimum number

Definition at line 1035 of file types.h.

8.5.2.77 VTSS_ACL_POLICY_NO_MAX

```
#define VTSS_ACL_POLICY_NO_MAX 255
```

ACLs policy maximum number

Definition at line 1037 of file types.h.

8.5.2.78 VTSS_ACL_POLICIES

```
#define VTSS_ACL_POLICIES (VTSS_ACL_POLICY_NO_MAX + 1)
```

Number of ACL policies

Definition at line 1043 of file types.h.

8.5.2.79 VTSS_ACL_POLICY_NO_START

```
#define VTSS_ACL_POLICY_NO_START VTSS_ACL_POLICY_NO_MIN
```

ACL policy start number

Definition at line 1044 of file types.h.

8.5.2.80 VTSS_ACL_POLICY_NO_END

```
#define VTSS_ACL_POLICY_NO_END (VTSS_ACL_POLICY_NO_START + VTSS_ACL_POLICIES)
```

ACL policy end number

Definition at line 1045 of file types.h.

8.5.2.81 VTSS_HQOS_COUNT

```
#define VTSS_HQOS_COUNT 256
```

Maximum number of HQoS entries

Definition at line 1165 of file types.h.

8.5.2.82 VTSS_HQOS_ID_NONE

```
#define VTSS_HQOS_ID_NONE 0xffff
```

Special HQoS ID value

Definition at line 1167 of file types.h.

8.5.2.83 VTSS_ONE_MIA

```
#define VTSS_ONE_MIA 1000000000
```

One billion

Definition at line 1198 of file types.h.

8.5.2.84 VTSS_ONE_MILL

```
#define VTSS_ONE_MILL 1000000
```

One million

Definition at line 1199 of file types.h.

8.5.2.85 VTSS_MAX_TIMEINTERVAL

```
#define VTSS_MAX_TIMEINTERVAL 0x7fffffffffffffLL
```

Maximum time interval

Definition at line 1200 of file types.h.

8.5.2.86 VTSS_INTERVAL_SEC

```
#define VTSS_INTERVAL_SEC( t ) ((i32)VTSS_DIV64((t)>>16, VTSS_ONE_MIA))
```

One Second time interval

Definition at line 1202 of file types.h.

8.5.2.87 VTSS_INTERVAL_MS

```
#define VTSS_INTERVAL_MS( t ) ((i32)VTSS_DIV64((t)>>16, VTSS_ONE_MILL))
```

One millisecond time interval

Definition at line 1203 of file types.h.

8.5.2.88 VTSS_INTERVAL_US

```
#define VTSS_INTERVAL_US( t ) ((i32)VTSS_DIV64((t)>>16, 1000))
```

One microsecond time interval

Definition at line 1204 of file types.h.

8.5.2.89 VTSS_INTERVAL_NS

```
#define VTSS_INTERVAL_NS( t ) ((i32)VTSS_MOD64((t)>>16, VTSS_ONE_MIA))
```

This returns the ns part of the interval, not the total number of ns

Definition at line 1205 of file types.h.

8.5.2.90 VTSS_INTERVAL_PS

```
#define VTSS_INTERVAL_PS(  
    t ) (((((i32)(t & 0xffff))*1000)+0x8000)/0x10000)
```

This returns the ps part of the interval, not the total number of ps

Definition at line 1206 of file types.h.

8.5.2.91 VTSS_SEC_NS_INTERVAL

```
#define VTSS_SEC_NS_INTERVAL(  
    s,  
    n ) (((vtss_timeinterval_t)(n)+(vtss_timeinterval_t)(s)*VTSS_ONE_MIA)<<16)
```

TBD

Definition at line 1207 of file types.h.

8.5.2.92 VTSS_CLOCK_IDENTITY_LENGTH

```
#define VTSS_CLOCK_IDENTITY_LENGTH 8
```

Length of unique PTP identifier

Definition at line 1218 of file types.h.

8.5.2.93 VTSS_SYNCE_CLK_PORT_ARRAY_SIZE

```
#define VTSS_SYNCE_CLK_PORT_ARRAY_SIZE 2
```

SYNCE clock out port numberarray size

Definition at line 1232 of file types.h.

8.5.3 Typedef Documentation

8.5.3.1 i8

```
typedef char i8
```

Fallback Integer types.

8-bit signed

Definition at line 68 of file types.h.

8.5.3.2 i16

```
typedef signed short i16
```

16-bit signed

Definition at line 69 of file types.h.

8.5.3.3 i32

```
typedef signed int i32
```

32-bit signed

Definition at line 70 of file types.h.

8.5.3.4 i64

```
typedef signed long long i64
```

64-bit signed

Definition at line 71 of file types.h.

8.5.3.5 u8

```
typedef unsigned char u8
```

8-bit unsigned

Definition at line 73 of file types.h.

8.5.3.6 u16

```
typedef unsigned short u16
```

16-bit unsigned

Definition at line 74 of file types.h.

8.5.3.7 u32

```
typedef unsigned int u32
```

32-bit unsigned

Definition at line 75 of file types.h.

8.5.3.8 u64

```
typedef unsigned long long u64
```

64-bit unsigned

Definition at line 76 of file types.h.

8.5.3.9 BOOL

```
typedef unsigned char BOOL
```

Boolean implemented as 8-bit unsigned

Definition at line 78 of file types.h.

8.5.3.10 uintptr_t

```
typedef unsigned int uintptr_t
```

Unsigned integer big enough to hold pointers

Definition at line 79 of file types.h.

8.5.3.11 vtss_mac_addr_t

```
typedef u8 vtss_mac_addr_t[VTSS_MAC_ADDR_SZ_BYTES]
```

MAC address (SMAC/DMAC)

Definition at line 649 of file types.h.

8.5.3.12 vtss_isdx_t

```
typedef u32 vtss_isdx_t
```

Ingress Service Index type

Definition at line 673 of file types.h.

8.5.3.13 vtss_packet_rx_grp_t

```
typedef u32 vtss_packet_rx_grp_t
```

Description: CPU Rx group number.

This is a value in range [0; VTSS_PACKET_RX_GRP_CNT[.

Definition at line 711 of file types.h.

8.5.3.14 vtss_packet_tx_grp_t

```
typedef u32 vtss_packet_tx_grp_t
```

Description: CPU Tx group number.

This is a value in range [0; VTSS_PACKET_TX_GRP_CNT[.

Definition at line 716 of file types.h.

8.5.4 Enumeration Type Documentation

8.5.4.1 anonymous enum

```
anonymous enum
```

Error codes.

Enumerator

VTSS_RC_OK	Success
VTSS_RC_ERROR	Unspecified error
VTSS_RC_INV_STATE	Invalid state for operation
VTSS_RC_INCOMPLETE	Incomplete result
VTSS_RC_ERR_CLK_CONF_NOT_SUPPORTED	The PHY doesn't support clock configuration (for SyncE)
VTSS_RC_ERR_KR_CONF_NOT_SUPPORTED	The PHY doesn't support 10GBASE_KR equalization
VTSS_RC_ERR_KR_CONF_INVALID_PARAMETER	One of the parameters are out of range
VTSS_RC_ERR_PHY_BASE_NO_NOT_FOUND	Port base number (first port within a chip) is not found
VTSS_RC_ERR_PHY_6G_MACRO_SETUP	Setup of 6G macro failed
VTSS_RC_ERR_PHY_MEDIA_IF_NOT_SUPPORTED	PHY does not support the selected media mode
VTSS_RC_ERR_PHY_CLK_CONF_NOT_SUPPORTED	The PHY doesn't support clock configuration (for SyncE)
VTSS_RC_ERR_PHY_GPIO_ALT_MODE_NOT_SUPPORTED	The PHY doesn't support the alternative mode for the selected GPIO pin
VTSS_RC_ERR_PHY_GPIO_PIN_NOT_SUPPORTED	The PHY doesn't support the selected GPIO pin
VTSS_RC_ERR_PHY_PORT_OUT_RANGE	PHY API called with port number larger than VTSS_PORTS
VTSS_RC_ERR_PHY_PATCH_SETTING_NOT_SUPPORTED	PHY API micro patch setting not supported for the port in question
VTSS_RC_ERR_PHY_LCPLL_NOT_SUPPORTED	PHY API LC-PLL status not supported for the port
VTSS_RC_ERR_PHY_RCPLL_NOT_SUPPORTED	PHY API RC-PLL status not supported for the port
VTSS_RC_ERR_MACSEC_INVALID_SCI_MACADDR	From IEEE 802.1AE-2006, section 9.9 - The 64-bit value FF-FF-FF-FF-FF-FF is never used as an SCI and is reserved for use by implementations to indicate the absence of an SC or an SCI in contexts where an SC can be present
VTSS_RC_ERR_MACSEC_NOT_ENABLED	Trying to access port where MACSEC is not enabled
VTSS_RC_ERR_MACSEC_SECY_ALREADY_IN_USE	Trying to use a secy which is already in use
VTSS_RC_ERR_MACSEC_NO_SECY_FOUND	No SecY found for the specific port
VTSS_RC_ERR_MACSEC_NO_SECY_VACANCY	No secy vacancy
VTSS_RC_ERR_MACSEC_INVALID_VALIDATE_FRAME	Validate_frames value invalid
VTSS_RC_ERR_MACSEC_COULD_NOT_PRG_SA_MATCH	Could not program the SA match
VTSS_RC_ERR_MACSEC_COULD_NOT_PRG_SA_FLOW	Could not program the SA flow
VTSS_RC_ERR_MACSEC_COULD_NOT_ENA_SA	Could not enable the SA
VTSS_RC_ERR_MACSEC_COULD_NOT_SET_SA	Could not set SA to in use
VTSS_RC_ERR_MACSEC_INVALID_BYPASS_HEADER_LENGTH	Invalid header bypass length
VTSS_RC_ERR_MACSEC_SC_NOT_FOUND	Could not find SC (from sci)
VTSS_RC_ERR_MACSEC_NO_CTRL_FRAME_MATCH	No control frame match
VTSS_RC_ERR_MACSEC_COULD_NOT_SET_PATTERN	Could no set bypass pattern for CP rule
VTSS_RC_ERR_MACSEC_TIMEOUT_ISSUE	Internal timeout issue, bailing out

Enumerator

VTSS_RC_ERR_MACSEC_COULD_NOT_EMPT← Y_EGRESS	Could not empty the egress pipeline
VTSS_RC_ERR_MACSEC_AN_NOT_CREATED	AN not created.
VTSS_RC_ERR_MACSEC_COULD_NOT_EMPT← Y_INGRESS	Could not empty the ingress pipeline
VTSS_RC_ERR_MACSEC_TX_SC_NOT_EXIST	No tx SC found
VTSS_RC_ERR_MACSEC_COULD_NOT_DISAB← LE_SA	Could not disable sa
VTSS_RC_ERR_MACSEC_COULD_NOT_DEL_R← X_SA	Could not delete rx sa
VTSS_RC_ERR_MACSEC_COULD_NOT_DEL_T← X_SA	Could not delete tx sa
VTSS_RC_ERR_MACSEC_PATTERN_NOT_SET	Pattern not set
VTSS_RC_ERR_MACSEC_HW_RESOURCE_EX← HUSTED	HW resources exhausted
VTSS_RC_ERR_MACSEC_SCI_ALREADY_EXISTS	SCI already exists
VTSS_RC_ERR_MACSEC_SC_RESOURCE_NO← T_FOUND	Could not find SC resources
VTSS_RC_ERR_MACSEC_RX_AN_ALREADY_I← N_USE	Rx AN is in use
VTSS_RC_ERR_MACSEC_EMPTY_RECORD	Could not get an empty record
VTSS_RC_ERR_MACSEC_COULD_NOT_PRG← XFORM	Could not program the xform record
VTSS_RC_ERR_MACSEC_COULD_NOT_TOGG← LE_SA	Could not toggle SA
VTSS_RC_ERR_MACSEC_TX_AN_ALREADY_I← N_USE	Tx AN is in use
VTSS_RC_ERR_MACSEC_ALL_AVAILABLE_SA← _IN_USE	All available SA's are in use
VTSS_RC_ERR_MACSEC_MATCH_DISABLE	MACSEC match disabled
VTSS_RC_ERR_MACSEC_ALL_CP_RULES_IN← USE	All CP rules of the specific type are in use
VTSS_RC_ERR_MACSEC_PATTERN_PRIO_NO← T_VALID	The pattern priority is not valid
VTSS_RC_ERR_MACSEC_BUFFER_TOO_SMALL	Buffer to small, must be greater than VTSS_MACSEC_FRAME_CAPTURE_SIZE_MAX
VTSS_RC_ERR_MACSEC_FRAME_TOO_LONG	Frame length is supposed to be less than the amount of data in the fifo
VTSS_RC_ERR_MACSEC_FRAME_TRUNCATED	Frame is Truncated
VTSS_RC_ERR_MACSEC_PHY_POWERED_DO← WN	Phy is powered down, i.e. the MacSec block is not accessible
VTSS_RC_ERR_MACSEC_PHY_NOT_MACSEC← _CAPABLE	Port/Phy is not MacSec capable
VTSS_RC_ERR_MACSEC_AN_NOT_EXIST	AN does not exist
VTSS_RC_ERR_MACSEC_NO_PATTERN_CFG	No pattern is configured
VTSS_RC_ERR_MACSEC_MAX_MTU	Maximum MTU allowed is 32761 (+ 4 bytes for VLAN)
VTSS_RC_ERR_MACSEC_UNEXPECT_CP_MODE	Unexpected CP mode
VTSS_RC_ERR_MACSEC_COULD_NOT_DISAB← LE_AN	Could not disable AN
VTSS_RC_ERR_MACSEC_RULE_OUT_OF_RAN← GE	Rule id is out of range. Must not be larger than VTSS_MACSEC_CP_RULES

Enumerator

VTSS_RC_ERR_MACSEC_RULE_NOT_EXIST	Rule does not exist
VTSS_RC_ERR_MACSEC_CSR_READ	Could not do CSR read
VTSS_RC_ERR_MACSEC_CSR_WRITE	Could not do CSR write
VTSS_RC_ERR_PHY_6G_RCPLL_ON_BASE_P←_ORT_ONLY	PHY API 6G RC-PLL status support only on Base port
VTSS_RC_ERR_INVALID_NULL_PTR	A pointer was unexpected NULL

Definition at line 139 of file types.h.

8.5.4.2 vtss_mem_flags_t

```
enum vtss_mem_flags_t
```

Memory allocation flags.

The VTSS API asks the application to allocate dynamic memory for its internal structures through calls to [VTSS_OS_MALLOC\(\)](#).

The application should normally just associate this with a call to malloc() or kmalloc() depending on the OS and the runtime model (API running in Kernel or User space).

However, on some OSs, it's required to allocate specially if the memory is going to be associated with DMA, hence the VTSS_MEM_FLAGS_DMA enumeration.

Also, to be able to support warm restart, another enumeration, VTSS_MEM_FLAGS_PERSIST, tells the application to allocate the memory in a part of RAM that won't be affected by a subsequent boot.

[VTSS_OS_MALLOC\(\)](#) must not block or make waiting points if called with flags != VTSS_MEM_FLAGS_NONE.

Each of the enumerations are ORed together to form the final flags that are used in a call to [VTSS_OS_MALLOC\(\)](#).

The same set of flags are used in calls to [VTSS_OS_FREE\(\)](#).

Enumerator

VTSS_MEM_FLAGS_NONE	Allocate normally according to runtime model (User or Kernel space).
VTSS_MEM_FLAGS_DMA	Allocate memory that can be used with a DMA.
VTSS_MEM_FLAGS_PERSIST	Allocate memory that will survive a warm restart.

Definition at line 275 of file types.h.

8.5.4.3 vtss_port_interface_t

```
enum vtss_port_interface_t
```

The different interfaces for connecting MAC and PHY.

Enumerator

VTSS_PORT_INTERFACE_NO_CONNECTION	No connection
VTSS_PORT_INTERFACE_LOOPBACK	Internal loopback in MAC
VTSS_PORT_INTERFACE_INTERNAL	Internal interface
VTSS_PORT_INTERFACE_MII	MII (RMII does not exist)
VTSS_PORT_INTERFACE_GMII	GMII
VTSS_PORT_INTERFACE_RGMII	RGMII
VTSS_PORT_INTERFACE_TBI	TBI
VTSS_PORT_INTERFACE_RTBI	RTBI
VTSS_PORT_INTERFACE_SGMII	SGMII
VTSS_PORT_INTERFACE_SGMII_CISCO	SGMII using Cisco aneg
VTSS_PORT_INTERFACE_SERDES	SERDES
VTSS_PORT_INTERFACE_VAUI	VAUI
VTSS_PORT_INTERFACE_100FX	100FX
VTSS_PORT_INTERFACE_XAUI	XAUI
VTSS_PORT_INTERFACE_RXAUI	RXAUI
VTSS_PORT_INTERFACE_XGMII	XGMII
VTSS_PORT_INTERFACE_SPI4	SPI4
VTSS_PORT_INTERFACE_QSGMII	QSGMII
VTSS_PORT_INTERFACE_SFI	SFI/LAN

Definition at line 457 of file types.h.

8.5.4.4 vtss_serdes_mode_t

```
enum vtss_serdes_mode_t
```

Serdes macro mode.

Enumerator

VTSS_SERDES_MODE_DISABLE	Disable serdes
VTSS_SERDES_MODE_XAUI_12G	XAUI 12G mode
VTSS_SERDES_MODE_XAUI	XAUI 10G mode
VTSS_SERDES_MODE_RXAUI	RXAUI 10G mode
VTSS_SERDES_MODE_RXAUI_12G	RXAUI 12G mode
VTSS_SERDES_MODE_2G5	2.5G mode
VTSS_SERDES_MODE_QSGMII	QSGMII mode
VTSS_SERDES_MODE_SGMII	SGMII mode
VTSS_SERDES_MODE_100FX	100FX mode
VTSS_SERDES_MODE_1000BaseX	1000BaseX mode
VTSS_SERDES_MODE_SFI	LAN/10G mode
VTSS_SERDES_MODE_SFI_DAC	LAN/10G DAC(CU)
VTSS_SERDES_MODE_IDLE	Send idles

Definition at line 490 of file types.h.

8.5.4.5 vtss_storm_policer_mode_t

enum `vtss_storm_policer_mode_t`

Storm policer mode configuration.

Enumerator

VTSS_STORM_POLICER_MODE_PORTS_AND_CPU	Police both CPU and front port destinations
VTSS_STORM_POLICER_MODE_PORTS_ONLY	Police front port destinations only
VTSS_STORM_POLICER_MODE_CPU_ONLY	Police CPU destination only

Definition at line 572 of file types.h.

8.5.4.6 vtss_policer_type_t

enum `vtss_policer_type_t`

Dual leaky buckets policer configuration.

Enumerator

VTSS_POLICER_TYPE_MEF	MEF bandwidth profile
VTSS_POLICER_TYPE_SINGLE	Single bucket policer (CIR/CBS)

Definition at line 587 of file types.h.

8.5.4.7 vtss_vlan_frame_t

enum `vtss_vlan_frame_t`

VLAN acceptable frame type.

Enumerator

VTSS_VLAN_FRAME_ALL	Accept all frames
VTSS_VLAN_FRAME_TAGGED	Accept tagged frames only
VTSS_VLAN_FRAME_UNTAGGED	Accept untagged frames only

Definition at line 618 of file types.h.

8.5.4.8 vtss_packet_reg_type_t

enum `vtss_packet_reg_type_t`

Packet registration type.

Enumerator

<code>VTSS_PACKET_REG_NORMAL</code>	Global registration configuration is used
<code>VTSS_PACKET_REG_FORWARD</code>	Forward normally
<code>VTSS_PACKET_REG_DISCARD</code>	Discard
<code>VTSS_PACKET_REG_CPU_COPY</code>	Copy to CPU
<code>VTSS_PACKET_REG_CPU_ONLY</code>	Redirect to CPU

Definition at line 753 of file types.h.

8.5.4.9 vtss_vdd_t

enum `vtss_vdd_t`

VDD power supply.

Enumerator

<code>VTSS_VDD_1V0</code>	1.0V (default)
<code>VTSS_VDD_1V2</code>	1.2V

Definition at line 776 of file types.h.

8.5.4.10 vtss_ip_type_t

enum `vtss_ip_type_t`

IP address type.

Enumerator

<code>VTSS_IP_TYPE_NONE</code>	Matches "InetAddressType_unknown"
<code>VTSS_IP_TYPE_IPV4</code>	Matches "InetAddressType_ipv4"
<code>VTSS_IP_TYPE_IPV6</code>	Matches "InetAddressType_ipv6"

Definition at line 806 of file types.h.

8.5.4.11 vtss_vcap_bit_t

enum `vtss_vcap_bit_t`

VCAP 1 bit.

Enumerator

VTSS_VCAP_BIT_ANY	Value 0 or 1
VTSS_VCAP_BIT_0	Value 0
VTSS_VCAP_BIT_1	Value 1

Definition at line 904 of file types.h.

8.5.4.12 vtss_vcap_vr_type_t

enum `vtss_vcap_vr_type_t`

Value/Range type.

Enumerator

VTSS_VCAP_VR_TYPE_VALUE_MASK	Used as value/mask
VTSS_VCAP_VR_TYPE_RANGE_INCLUSIVE	Used as inclusive range: low <= range <= high
VTSS_VCAP_VR_TYPE_RANGE_EXCLUSIVE	Used as exclusive range: range < low or range > high

Definition at line 983 of file types.h.

8.5.4.13 vtss_vcap_key_type_t

enum `vtss_vcap_key_type_t`

VCAP key type.

Enumerator

VTSS_VCAP_KEY_TYPE_NORMAL	Half key, SIP only
VTSS_VCAP_KEY_TYPE_DOUBLE_TAG	Quarter key, two tags
VTSS_VCAP_KEY_TYPE_IP_ADDR	Half key, SIP and DIP
VTSS_VCAP_KEY_TYPE_MAC_IP_ADDR	Full key, MAC and IP addresses

Definition at line 1013 of file types.h.

8.5.4.14 vtss_ece_dir_t

enum `vtss_ece_dir_t`

ECE direction.

Enumerator

VTSS_ECE_DIR_BOTH	Bidirectional
VTSS_ECE_DIR_UNI_TO_NNI	UNI-to-NNI direction
VTSS_ECE_DIR_NNI_TO_UNI	NNI-to-UNI direction

Definition at line 1058 of file types.h.

8.5.4.15 vtss_ece_pop_tag_t

enum `vtss_ece_pop_tag_t`

Ingress tag popping.

Enumerator

VTSS_ECE_POP_TAG_0	No tag popping
VTSS_ECE_POP_TAG_1	Pop one tag
VTSS_ECE_POP_TAG_2	Pop two tags (VTSS_ECE_DIR_NNI_TO_UNI only)

Definition at line 1066 of file types.h.

8.5.4.16 vtss_ece_inner_tag_type_t

enum `vtss_ece_inner_tag_type_t`

ECE inner tag type.

Enumerator

VTSS_ECE_INNER_TAG_NONE	No inner tag
VTSS_ECE_INNER_TAG_C	Inner tag is C-tag
VTSS_ECE_INNER_TAG_S	Inner tag is S-tag
VTSS_ECE_INNER_TAG_S_CUSTOM	Inner tag is S-custom tag

Definition at line 1118 of file types.h.

8.5.4.17 vtss_hqos_sch_mode_t

enum `vtss_hqos_sch_mode_t`

HQoS port scheduling mode.

The scheduling mode for the port affects which egress QoS options are available.

Enumerator

VTSS_HQOS_SCH_MODE_NORMAL	Normal QoS configuration available for non-service traffic only (default)
VTSS_HQOS_SCH_MODE_BASIC	Basic QoS configuration available for non-service traffic only
VTSS_HQOS_SCH_MODE_HIERARCHICAL	Basic QoS configuration available per HQoS entry (HQoS)

Definition at line 1173 of file types.h.

8.6 vtss_api/include/vtss_ae_api.h File Reference

ae API

```
#include <vtss/api/types.h>
```

8.6.1 Detailed Description

ae API

8.7 vtss_api/include/vtss_afi_api.h File Reference

AFI API.

```
#include <vtss/api/options.h>
```

8.7.1 Detailed Description

AFI API.

This header file describes Automatic Frame Injector functions.

8.8 vtss_api/include/vtss_aneg_api.h File Reference

ANEG API.

```
#include <vtss/api/types.h>
```

8.8.1 Detailed Description

ANEG API.

8.9 vtss_api/include/vtss_api.h File Reference

Vitesse API main header file.

```
#include <vtss/api/options.h>
#include <vtss_os.h>
#include <vtss/api/types.h>
#include <vtss_init_api.h>
#include <vtss_misc_api.h>
#include <vtss_port_api.h>
#include <vtss_phy_api.h>
#include <vtss_phy_10g_api.h>
#include <vtss_qos_api.h>
#include <vtss_packet_api.h>
#include <vtss_security_api.h>
#include <vtss_l2_api.h>
#include <vtss_l3_api.h>
#include <vtss_evc_api.h>
#include <vtss_sync_api.h>
#include <vtss_phy_ts_api.h>
#include <vtss_ts_api.h>
#include <vtss_wis_api.h>
```

8.9.1 Detailed Description

Vitesse API main header file.

This is the only header file which must be included by the application

8.10 vtss_api/include/vtss_evc_api.h File Reference

EVC API.

```
#include <vtss/api/types.h>
```

Data Structures

- struct [vtss_evc_port_conf_t](#)
EVC port configuration.
- struct [vtss_evc_pb_conf_t](#)
PB specific EVC configuration.
- struct [vtss_evc_conf_t](#)
EVC configuration (excluding UNIs)
- struct [vtss_ece_mac_t](#)
ECE MAC information.
- struct [vtss_ece_tag_t](#)
ECE tag information.
- struct [vtss_ece_frame_ipv4_t](#)
ECE IPv4 information.
- struct [vtss_ece_frame_ipv6_t](#)
ECE IPv6 information.
- struct [vtss_ece_key_t](#)
ECE key.
- struct [vtss_ece_outer_tag_t](#)
ECE outer tag.
- struct [vtss_ece_inner_tag_t](#)
ECE inner tag.
- struct [vtss_ece_action_t](#)
ECE action.
- struct [vtss_ece_t](#)
EVC Control Entry.

Macros

- #define VTSS_EVC_POLICERS 2048
- #define VTSS_EVC_POLICER_ID_DISCARD 4094
- #define VTSS_EVC_POLICER_ID_NONE 4095
- #define VTSS_EVC_POLICER_ID_EVC 4096
- #define VTSS_EVC_ID_NONE 0xffff
- #define VTSS_ECE_ID_LAST 0

Typedefs

- typedef [vtss_dlb_policer_conf_t](#) [vtss_evc_policer_conf_t](#)
EVC policer configuration.

Enumerations

- enum [vtss_ece_type_t](#) { VTSS_ECE_TYPE_ANY, VTSS_ECE_TYPE_IPV4, VTSS_ECE_TYPE_IPV6 }
ECE frame type.
- enum [vtss_ece_port_t](#) { VTSS_ECE_PORT_NONE, VTSS_ECE_PORT_ROOT }
ECE port type.

Functions

- `vtss_rc vtss_evc_port_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_evc_port_conf_t` *const conf)

Get EVC port configuration.
- `vtss_rc vtss_evc_port_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_evc_port_conf_t` *const conf)

Set EVC port configuration.
- `vtss_rc vtss_evc_policer_conf_get` (const `vtss_inst_t` inst, const `vtss_evc_policer_id_t` policer_id, `vtss_evc_policer_conf_t` *const conf)

Get EVC policer configuration.
- `vtss_rc vtss_evc_policer_conf_set` (const `vtss_inst_t` inst, const `vtss_evc_policer_id_t` policer_id, const `vtss_evc_policer_conf_t` *const conf)

Set EVC policer configuration.
- `vtss_rc vtss_evc_add` (const `vtss_inst_t` inst, const `vtss_evc_id_t` evc_id, const `vtss_evc_conf_t` *const conf)

Add EVC.
- `vtss_rc vtss_evc_del` (const `vtss_inst_t` inst, const `vtss_evc_id_t` evc_id)

Delete EVC.
- `vtss_rc vtss_evc_get` (const `vtss_inst_t` inst, const `vtss_evc_id_t` evc_id, `vtss_evc_conf_t` *const conf)

Get EVC configuration.
- `vtss_rc vtss_ece_init` (const `vtss_inst_t` inst, const `vtss_ece_type_t` type, `vtss_ece_t` *const ece)

Initialize ECE to default values.
- `vtss_rc vtss_ece_add` (const `vtss_inst_t` inst, const `vtss_ece_id_t` ece_id, const `vtss_ece_t` *const ece)

Add/modify ECE.
- `vtss_rc vtss_ece_del` (const `vtss_inst_t` inst, const `vtss_ece_id_t` ece_id)

Delete ECE.
- `vtss_rc vtss_evc_counters_get` (const `vtss_inst_t` inst, const `vtss_evc_id_t` evc_id, const `vtss_port_no_t` port_no, `vtss_evc_counters_t` *const counters)

Get EVC counters for a port.
- `vtss_rc vtss_evc_counters_clear` (const `vtss_inst_t` inst, const `vtss_evc_id_t` evc_id, const `vtss_port_no_t` port_no)

Clear EVC counters for a port.
- `vtss_rc vtss_ece_counters_get` (const `vtss_inst_t` inst, const `vtss_ece_id_t` ece_id, const `vtss_port_no_t` port_no, `vtss_evc_counters_t` *const counters)

Get ECE counters for a port.
- `vtss_rc vtss_ece_counters_clear` (const `vtss_inst_t` inst, const `vtss_ece_id_t` ece_id, const `vtss_port_no_t` port_no)

Clear ECE counters for a port.

8.10.1 Detailed Description

EVC API.

This header file describes EVC functions

8.10.2 Macro Definition Documentation

8.10.2.1 VTSS_EVC_POLICERS

```
#define VTSS_EVC_POLICERS 2048
```

Maximum number of EVC policers

Definition at line 195 of file vtss_evc_api.h.

8.10.2.2 VTSS_EVC_POLICER_ID_DISCARD

```
#define VTSS_EVC_POLICER_ID_DISCARD 4094
```

EVC/ECE: Policer discards all frames

Definition at line 205 of file vtss_evc_api.h.

8.10.2.3 VTSS_EVC_POLICER_ID_NONE

```
#define VTSS_EVC_POLICER_ID_NONE 4095
```

EVC/ECE: Policer forwards all frames

Definition at line 206 of file vtss_evc_api.h.

8.10.2.4 VTSS_EVC_POLICER_ID_EVC

```
#define VTSS_EVC_POLICER_ID_EVC 4096
```

ECE only: Use EVC policer

Definition at line 207 of file vtss_evc_api.h.

8.10.2.5 VTSS_EVC_ID_NONE

```
#define VTSS_EVC_ID_NONE 0xfffff
```

Special EVC ID value

Definition at line 243 of file vtss_evc_api.h.

8.10.2.6 VTSS_ECE_ID_LAST

```
#define VTSS_ECE_ID_LAST 0
```

Special value used to add last in list

Definition at line 363 of file vtss_evc_api.h.

8.10.3 Enumeration Type Documentation

8.10.3.1 vtss_ece_type_t

```
enum vtss_ece_type_t
```

ECE frame type.

Enumerator

VTSS_ECE_TYPE_ANY	Any frame type
VTSS_ECE_TYPE_IPV4	IPv4
VTSS_ECE_TYPE_IPV6	IPv6

Definition at line 400 of file vtss_evc_api.h.

8.10.3.2 vtss_ece_port_t

```
enum vtss_ece_port_t
```

ECE port type.

Enumerator

VTSS_ECE_PORT_NONE	Port not included
VTSS_ECE_PORT_ROOT	Root UNI port

Definition at line 413 of file vtss_evc_api.h.

8.10.4 Function Documentation

8.10.4.1 vtss_evc_port_conf_get()

```
vtss_rc vtss_evc_port_conf_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_evc_port_conf_t *const conf )
```

Get EVC port configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>conf</i>	[OUT] EVC port configuration structure.

Returns

Return code.

8.10.4.2 vtss_evc_port_conf_set()

```
vtss_rc vtss_evc_port_conf_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_evc_port_conf_t *const conf )
```

Set EVC port configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>conf</i>	[IN] EVC port configuration structure.

Returns

Return code.

8.10.4.3 vtss_evc_policer_conf_get()

```
vtss_rc vtss_evc_policer_conf_get (
    const vtss_inst_t inst,
    const vtss_evc_policer_id_t policer_id,
    vtss_evc_policer_conf_t *const conf )
```

Get EVC policer configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>policer_id</i>	[IN] Policer ID.
<i>conf</i>	[OUT] Policer configuration.

Returns

Return code.

8.10.4.4 vtss_evc_policer_conf_set()

```
vtss_rc vtss_evc_policer_conf_set (
    const vtss_inst_t inst,
    const vtss_evc_policer_id_t policer_id,
    const vtss_evc_policer_conf_t *const conf )
```

Set EVC policer configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>policer_id</i>	[IN] Policer ID.
<i>conf</i>	[IN] Policer configuration.

Returns

Return code.

8.10.4.5 vtss_evc_add()

```
vtss_rc vtss_evc_add (
    const vtss_inst_t inst,
    const vtss_evc_id_t evc_id,
    const vtss_evc_conf_t *const conf )
```

Add EVC.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>evc_id</i>	[IN] EVC ID.
<i>conf</i>	[IN] EVC configuration.

Returns

Return code.

8.10.4.6 vtss_evc_del()

```
vtss_rc vtss_evc_del (
    const vtss_inst_t inst,
    const vtss_evc_id_t evc_id )
```

Delete EVC.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>evc_id</i>	[IN] EVC ID.

Returns

Return code.

8.10.4.7 vtss_evc_get()

```
vtss_rc vtss_evc_get (
    const vtss_inst_t inst,
    const vtss_evc_id_t evc_id,
    vtss_evc_conf_t *const conf )
```

Get EVC configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>evc_id</i>	[IN] EVC ID.
<i>conf</i>	[OUT] EVC configuration.

Returns

Return code.

8.10.4.8 vtss_ece_init()

```
vtss_rc vtss_ece_init (
    const vtss_inst_t inst,
    const vtss_ece_type_t type,
    vtss_ece_t *const ece )
```

Initialize ECE to default values.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>type</i>	[IN] ECE type.
<i>ece</i>	[OUT] ECE structure.

Returns

Return code.

8.10.4.9 vtss_ece_add()

```
vtss_rc vtss_ece_add (
    const vtss_inst_t inst,
    const vtss_ece_id_t ece_id,
    const vtss_ece_t *const ece )
```

Add/modify ECE.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>ece_id</i>	[IN] ECE ID. The ECE will be added before the entry with this ID. VTSS_ECE_ID_LAST is reserved for inserting last.
<i>ece</i>	[IN] ECE structure.

Returns

Return code.

8.10.4.10 vtss_ece_del()

```
vtss_rc vtss_ece_del (
    const vtss_inst_t inst,
    const vtss_ece_id_t ece_id )
```

Delete ECE.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>ece_id</i>	[IN] ECE ID.

Returns

Return code.

8.10.4.11 vtss_evc_counters_get()

```
vtss_rc vtss_evc_counters_get (
    const vtss_inst_t inst,
    const vtss_evc_id_t evc_id,
    const vtss_port_no_t port_no,
    vtss_evc_counters_t *const counters )
```

Get EVC counters for a port.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>evc_id</i>	[IN] EVC ID.
<i>port_no</i>	[IN] Port number.
<i>counters</i>	[OUT] EVC counters.

Returns

Return code.

8.10.4.12 vtss_evc_counters_clear()

```
vtss_rc vtss_evc_counters_clear (
    const vtss_inst_t inst,
    const vtss_evc_id_t evc_id,
    const vtss_port_no_t port_no )
```

Clear EVC counters for a port.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>evc_id</i>	[IN] EVC ID.
<i>port_no</i>	[IN] Port number.

Returns

Return code.

8.10.4.13 vtss_ece_counters_get()

```
vtss_rc vtss_ece_counters_get (
    const vtss_inst_t inst,
    const vtss_ece_id_t ece_id,
    const vtss_port_no_t port_no,
    vtss_evc_counters_t *const counters )
```

Get ECE counters for a port.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>ece_id</i>	[IN] ECE ID.
<i>port_no</i>	[IN] Port number.
<i>counters</i>	[OUT] ECE counters.

Returns

Return code.

8.10.4.14 vtss_ece_counters_clear()

```
vtss_rc vtss_ece_counters_clear (
    const vtss_inst_t inst,
    const vtss_ece_id_t ece_id,
    const vtss_port_no_t port_no )
```

Clear ECE counters for a port.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>ece_id</i>	[IN] ECE ID.
<i>port_no</i>	[IN] Port number.

Returns

Return code.

8.11 vtss_api/include/vtss_fdma_api.h File Reference

Frame DMA API.

```
#include <vtss/api/types.h>
```

8.11.1 Detailed Description

Frame DMA API.

8.12 vtss_api/include/vtss_gfp_api.h File Reference

GFP API.

```
#include <vtss/api/types.h>
```

8.12.1 Detailed Description

GFP API.

8.13 vtss_api/include/vtss_hqos_api.h File Reference

HQoS API.

```
#include <vtss/api/types.h>
```

8.13.1 Detailed Description

HQoS API.

This header file describes Hierarchical Quality of Service (HQoS) functions

HQoS is enabled on a port when the port scheduling mode is set to hierarchical.

HQoS parameters are configured using HQoS IDs.

Traffic can be mapped to HQoS IDs by using the corresponding modules.

8.14 vtss_api/include/vtss_i2c_api.h File Reference

I2C API.

```
#include <vtss/api/types.h>
```

8.14.1 Detailed Description

I2C API.

8.15 vtss_api/include/vtss_init_api.h File Reference

Initialization API.

```
#include <vtss/api/types.h>
```

Data Structures

- struct [vtss_inst_create_t](#)
Create structure.
- struct [vtss_pi_conf_t](#)
PI configuration.
- struct [serdes_fields_t](#)
Serdes fields.
- struct [vtss_serdes_macro_conf_t](#)
Serdes macro configuration.
- struct [vtss_qs_conf_t](#)
Queue System settings.
- struct [vtss_init_conf_t](#)
Initialization configuration.
- struct [vtss_restart_status_t](#)
Restart status.

Macros

- #define [VTSS_I2C_NO_MULTIPLEXER](#) -1
- #define [VTSS_QS_CONF_MAX](#) 0xFFFFFFFF1
- #define [VTSS_QS_CONF_MIN](#) 0xFFFFFFFF2

Typedefs

- `typedef vtss_rc(* vtss_reg_read_t) (const vtss_chip_no_t chip_no, const u32 addr, u32 *const value)`
Register read function.
- `typedef vtss_rc(* vtss_reg_write_t) (const vtss_chip_no_t chip_no, const u32 addr, const u32 value)`
Register write function.
- `typedef vtss_rc(* vtss_i2c_read_t) (const vtss_port_no_t port_no, const u8 i2c_addr, const u8 addr, u8 *const data, const u8 cnt, const i8 i2c_clk_sel)`
I2C read function.
- `typedef vtss_rc(* vtss_i2c_write_t) (const vtss_port_no_t port_no, const u8 i2c_addr, u8 *const data, const u8 cnt, const i8 i2c_clk_sel)`
I2C write function.
- `typedef vtss_rc(* vtss_spi_read_write_t) (const vtss_inst_t inst, const vtss_port_no_t port_no, const u8 bit-size, u8 *const bitstream)`
SPI read/write function.
- `typedef vtss_rc(* vtss_spi_32bit_read_write_t) (const vtss_inst_t inst, vtss_port_no_t port_no, BOOL read, u8 dev, u16 reg_num, u32 *const data)`
SPI 32 bit read/write function.
- `typedef vtss_rc(* vtss_spi_64bit_read_write_t) (const vtss_inst_t inst, vtss_port_no_t port_no, BOOL read, u8 dev, u16 reg_num, u64 *const data)`
SPI 64 bit read/write function.
- `typedef vtss_rc(* vtss_miim_read_t) (const vtss_inst_t inst, const vtss_port_no_t port_no, const u8 addr, u16 *const value)`
MII management read function (IEEE 802.3 clause 22)
- `typedef vtss_rc(* vtss_miim_write_t) (const vtss_inst_t inst, const vtss_port_no_t port_no, const u8 addr, const u16 value)`
MII management write function (IEEE 802.3 clause 22)
- `typedef vtss_rc(* vtss_mmd_read_t) (const vtss_inst_t inst, const vtss_port_no_t port_no, const u8 mmd, const u16 addr, u16 *const value)`
MMD management read function (IEEE 802.3 clause 45)
- `typedef vtss_rc(* vtss_mmd_read_inc_t) (const vtss_inst_t inst, const vtss_port_no_t port_no, const u8 mmd, const u16 addr, u16 *const buf, u8 count)`
MMD management read increment function (IEEE 802.3 clause 45)
- `typedef vtss_rc(* vtss_mmd_write_t) (const vtss_inst_t inst, const vtss_port_no_t port_no, const u8 mmd, const u16 addr, const u16 value)`
MMD management write function (IEEE 802.3 clause 45)
- `typedef u16 vtss_version_t`
API version.

Enumerations

- `enum vtss_target_type_t {`
`VTSS_TARGET_CU_PHY, VTSS_TARGET_10G_PHY, VTSS_TARGET_SPARX_III_11 = 0x7414,`
`VTSS_TARGET_SERVAL_LITE = 0x7416,`
`VTSS_TARGET_SERVAL = 0x7418, VTSS_TARGET_SEVILLE = 0x9953, VTSS_TARGET_SPARX_III_10_UM`
`= 0x7420, VTSS_TARGET_SPARX_III_17_UM = 0x7421,`
`VTSS_TARGET_SPARX_III_25_UM = 0x7422, VTSS_TARGET_CARACAL_LITE = 0x7423, VTSS_TARGET_SPARX_III_10`
`= 0x7424, VTSS_TARGET_SPARX_III_18 = 0x7425,`
`VTSS_TARGET_SPARX_III_24 = 0x7426, VTSS_TARGET_SPARX_III_26 = 0x7427, VTSS_TARGET_SPARX_III_10_01`
`= 0x17424, VTSS_TARGET_CARACAL_1 = 0x7428,`
`VTSS_TARGET_CARACAL_2 = 0x7429, VTSS_TARGET_JAGUAR_1 = 0x7460, VTSS_TARGET_LYNX_1`
`= 0x7462, VTSS_TARGET_E_STAX_III_48 = 0x7432,`
`VTSS_TARGET_E_STAX_III_68 = 0x7434, VTSS_TARGET_E_STAX_III_24_DUAL = 0xD7431,`

```

VTSS_TARGET_E_STAX_III_68_DUAL = 0xD7434, VTSS_TARGET_DAYTONA = 0x8492,
VTSS_TARGET_TALLADEGA = 0x8494, VTSS_TARGET_SERVAL_2 = 0x7438, VTSS_TARGET_LYNX_2
= 0x7464, VTSS_TARGET_JAGUAR_2 = 0x7468,
VTSS_TARGET_SPARX_IV_52 = 0x7442, VTSS_TARGET_SPARX_IV_44 = 0x7444, VTSS_TARGET_SPARX_IV_80
= 0x7448, VTSS_TARGET_SPARX_IV_90 = 0x7449 }

```

Target chip type.

- enum `vtss_pi_width_t` { `VTSS_PI_WIDTH_16` = 0, `VTSS_PI_WIDTH_8` }

PI data width.

- enum `vtss_port_mux_mode_t` { `VTSS_PORT_MUX_MODE_0`, `VTSS_PORT_MUX_MODE_1`, `VTSS_PORT_MUX_MODE_7` }

Port mux configuration.

- enum `vtss_restart_info_src_t` { `VTSS_RESTART_INFO_SRC_NONE`, `VTSS_RESTART_INFO_SRC_CU_PHY`, `VTSS_RESTART_INFO_SRC_10G_PHY` }

Restart information source.

- enum `vtss_qs_mode_t` { `VTSS_QS_MODE_DISABLED`, `VTSS_QS_MODE_ENABLED` }

The major modes of QS.

- enum `vtss_restart_t` { `VTSS_RESTART_COLD`, `VTSS_RESTART_COOL`, `VTSS_RESTART_WARM` }

Restart type.

Functions

- `vtss_rc vtss_inst_get` (const `vtss_target_type_t` target, `vtss_inst_create_t` *const create)

Initialize create structure for target.

- `vtss_rc vtss_inst_create` (const `vtss_inst_create_t` *const create, `vtss_inst_t` *const inst)

Create target instance.

- `vtss_rc vtss_inst_destroy` (const `vtss_inst_t` inst)

Destroy target instance.

- `vtss_rc vtss_init_conf_get` (const `vtss_inst_t` inst, `vtss_init_conf_t` *const conf)

Get default initialization configuration.

- `vtss_rc vtss_init_conf_set` (const `vtss_inst_t` inst, const `vtss_init_conf_t` *const conf)

Set initialization configuration.

- `vtss_rc vtss_restart_conf_end` (const `vtss_inst_t` inst)

Indicate configuration end. If a warm start has been done, the stored configuration will be applied.

- `vtss_rc vtss_restart_status_get` (const `vtss_inst_t` inst, `vtss_restart_status_t` *const status)

Get restart status.

- `vtss_rc vtss_restart_conf_get` (const `vtss_inst_t` inst, `vtss_restart_t` *const restart)

Get restart configuration (next restart mode)

- `vtss_rc vtss_restart_conf_set` (const `vtss_inst_t` inst, const `vtss_restart_t` restart)

Set restart configuration (next restart mode)

- `vtss_rc vtss_qs_conf_set` (const `vtss_inst_t` inst, const `vtss_qs_conf_t` *const qs)

Configure the Queue System.

- `vtss_rc vtss_qs_conf_get` (const `vtss_inst_t` inst, `vtss_qs_conf_t` *const qs)

Get the configuration of the Queue System.

8.15.1 Detailed Description

Initialization API.

This header file describes functions used to create and initialize targets

8.15.2 Macro Definition Documentation

8.15.2.1 VTSS_I2C_NO_MULTIPLEXER

```
#define VTSS_I2C_NO_MULTIPLEXER -1
```

Used to signal not to use internal i2c clock multiplexing

Definition at line 139 of file vtss_init_api.h.

8.15.2.2 VTSS_QS_CONF_MAX

```
#define VTSS_QS_CONF_MAX 0xFFFFFFFF1
```

Configure QS to max possible value

Definition at line 395 of file vtss_init_api.h.

8.15.2.3 VTSS_QS_CONF_MIN

```
#define VTSS_QS_CONF_MIN 0xFFFFFFFF2
```

Configure QS to min value (guaranteed)

Definition at line 396 of file vtss_init_api.h.

8.15.3 Typedef Documentation

8.15.3.1 vtss_reg_read_t

```
typedef vtss_rc(* vtss_reg_read_t) (const vtss_chip_no_t chip_no, const u32 addr, u32 *const value)
```

Register read function.

Parameters

<i>chip_no</i>	[IN] Chip number, for targets with multiple chips
<i>addr</i>	[IN] Register address
<i>value</i>	[OUT] Register value

Returns

Return code.

Definition at line 122 of file vtss_init_api.h.

8.15.3.2 vtss_reg_write_t

```
typedef vtss_rc(* vtss_reg_write_t) (const vtss_chip_no_t chip_no, const u32 addr, const u32
value)
```

Register write function.

Parameters

<i>chip_no</i>	[IN] Chip number, for targets with multiple chips
<i>addr</i>	[IN] Register address
<i>value</i>	[IN] Register value

Returns

Return code.

Definition at line 135 of file vtss_init_api.h.

8.15.3.3 vtss_i2c_read_t

```
typedef vtss_rc(* vtss_i2c_read_t) (const vtss_port_no_t port_no, const u8 i2c_addr, const u8
addr, u8 *const data, const u8 cnt, const i8 i2c_clk_sel)
```

I2C read function.

Parameters

<i>port_no</i>	[IN] Port number
<i>i2c_addr</i>	[IN] I2C device address
<i>addr</i>	[IN] Register address
<i>data</i>	[OUT] Pointer the register(s) data value.
<i>cnt</i>	[IN] Number of registers to read
<i>i2c_clk_sel</i>	[IN] If i2c clock multiplexing is supported then this is the i2c mux, else use VTSS_I2C_NO_MULTIPLEXER

Returns

Return code.

Definition at line 152 of file vtss_init_api.h.

8.15.3.4 vtss_i2c_write_t

```
typedef vtss_rc(* vtss_i2c_write_t) (const vtss_port_no_t port_no, const u8 i2c_addr, u8 *const data, const u8 cnt, const i8 i2c_clk_sel)
```

I2C write function.

Parameters

<i>port_no</i>	[IN] Port number
<i>i2c_addr</i>	[IN] I2C device address
<i>data</i>	[OUT] Pointer the data to be written.
<i>cnt</i>	[IN] Number of data bytes to write
<i>i2c_clk_sel</i>	[IN] If i2c clock multiplexing is supported then this is the i2c mux, else use VTSS_I2C_NO_MULTIPLEXER

Returns

Return code.

Definition at line 170 of file vtss_init_api.h.

8.15.3.5 vtss_spi_read_write_t

```
typedef vtss_rc(* vtss_spi_read_write_t) (const vtss_inst_t inst, const vtss_port_no_t port_no, const u8 bitsize, u8 *const bitstream)
```

SPI read/write function.

Parameters

<i>inst</i>	[IN] Vitesse API instance.
<i>port_no</i>	[IN] Port number.
<i>bitsize</i>	[IN] Size (in bytes) of bitstream following this parameter.
<i>data</i>	[IN OUT] Pointer to the data to be written to SPI Slave, if doing write operation. Pointer to the data read from SPI Slave, if doing read operation.

Returns

Return code.

Definition at line 187 of file vtss_init_api.h.

8.15.3.6 vtss_spi_32bit_read_write_t

```
typedef vtss_rc(* vtss_spi_32bit_read_write_t) (const vtss_inst_t inst, vtss_port_no_t port_no, BOOL read, u8 dev, u16 reg_num, u32 *const data)
```

SPI 32 bit read/write function.

Parameters

<i>inst</i>	[IN] Vitesse API instance.
<i>port_no</i>	[IN] Port number.
<i>read</i>	[IN] Read/Write.
<i>dev</i>	[IN] MMD device number.
<i>reg_num</i>	[IN] Register offset.
<i>data</i>	[IN OUT] Pointer to the data to be written to SPI Slave, if doing write operation. Pointer to the data read from SPI Slave, if doing read operation.

Returns

Return code.

Definition at line 205 of file vtss_init_api.h.

8.15.3.7 vtss_spi_64bit_read_write_t

```
typedef vtss_rc(* vtss_spi_64bit_read_write_t) (const vtss_inst_t inst, vtss_port_no_t port_no, BOOL read, u8 dev, u16 reg_num, u64 *const data)
```

SPI 64 bit read/write function.

Parameters

<i>inst</i>	[IN] Vitesse API instance.
<i>port_no</i>	[IN] Port number.
<i>read</i>	[IN] Read/Write.
<i>dev</i>	[IN] MMD device number.
<i>reg_num</i>	[IN] Register offset.
<i>data</i>	[IN OUT] Pointer to the data to be written to SPI Slave, if doing write operation. Pointer to the data read from SPI Slave, if doing read operation.

Returns

Return code.

Definition at line 225 of file vtss_init_api.h.

8.15.3.8 vtss_miim_read_t

```
typedef vtss_rc(* vtss_miim_read_t) (const vtss_inst_t inst, const vtss_port_no_t port_no,  
const u8 addr, u16 *const value)
```

MII management read function (IEEE 802.3 clause 22)

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>addr</i>	[IN] Register address (0-31)
<i>value</i>	[OUT] Register value

Returns

Return code.

Definition at line 242 of file vtss_init_api.h.

8.15.3.9 vtss_miim_write_t

```
typedef vtss_rc(* vtss_miim_write_t) (const vtss_inst_t inst, const vtss_port_no_t port_no,  
const u8 addr, const u16 value)
```

MII management write function (IEEE 802.3 clause 22)

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>addr</i>	[IN] Register address (0-31)
<i>value</i>	[IN] Register value

Returns

Return code.

Definition at line 257 of file vtss_init_api.h.

8.15.3.10 vtss_mmd_read_t

```
typedef vtss_rc(* vtss_mmd_read_t) (const vtss_inst_t inst, const vtss_port_no_t port_no,  
const u8 mmd, const u16 addr, u16 *const value)
```

MMD management read function (IEEE 802.3 clause 45)

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>mmd</i>	[IN] MMD address (0-31)
<i>addr</i>	[IN] Register address (0-65535)
<i>value</i>	[OUT] Register value

Returns

Return code.

Definition at line 273 of file vtss_init_api.h.

8.15.3.11 vtss_mmd_read_inc_t

```
typedef vtss_rc(* vtss_mmd_read_inc_t) (const vtss_inst_t inst, const vtss_port_no_t port_no,
const u8 mmd, const u16 addr, u16 *const buf, u8 count)
```

MMD management read increment function (IEEE 802.3 clause 45)

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>mmd</i>	[IN] MMD address (0-31)
<i>addr</i>	[IN] Start register address (0-65535)
<i>buf</i>	[OUT] The register values (pointer provided by user)
<i>count</i>	[IN] Number of register reads (increment register reads)

Returns

Return code.

Definition at line 291 of file vtss_init_api.h.

8.15.3.12 vtss_mmd_write_t

```
typedef vtss_rc(* vtss_mmd_write_t) (const vtss_inst_t inst, const vtss_port_no_t port_no,
const u8 mmd, const u16 addr, const u16 value)
```

MMD management write function (IEEE 802.3 clause 45)

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>mmd</i>	[IN] MMD address (0-31)
<i>addr</i>	[IN] Start register address (0-65535)
<i>buf</i>	[IN] The register value

Returns

Return code.

Definition at line 309 of file vtss_init_api.h.

8.15.4 Enumeration Type Documentation

8.15.4.1 vtss_target_type_t

```
enum vtss_target_type_t
```

Target chip type.

Enumerator

VTSS_TARGET CU PHY	Cu PHY family
VTSS_TARGET 10G PHY	10G PHY family
VTSS_TARGET SPARX_III_11	SparX-III-11 SME switch
VTSS_TARGET SERVAL_LITE	Serval Lite CE switch
VTSS_TARGET SERVAL	Serval CE switch
VTSS_TARGET SEVILLE	Seville switch
VTSS_TARGET SPARX_III_10_UM	SparxIII-10 unmanaged switch
VTSS_TARGET SPARX_III_17_UM	SparxIII-17 unmanaged switch
VTSS_TARGET SPARX_III_25_UM	SparxIII-25 unmanaged switch
VTSS_TARGET CARACAL_LITE	Caracal-Lite CE switch
VTSS_TARGET SPARX_III_10	SparxIII-10 switch
VTSS_TARGET SPARX_III_18	SparxIII-18 switch
VTSS_TARGET SPARX_III_24	SparxIII-24 switch
VTSS_TARGET SPARX_III_26	SparxIII-26 switch
VTSS_TARGET SPARX_III_10_01	SparxIII-10-01 switch
VTSS_TARGET CARACAL_1	Caracal-1 CE switch
VTSS_TARGET CARACAL_2	Caracal-2 CE switch
VTSS_TARGET JAGUAR_1	Jaguar-1 CE switch
VTSS_TARGET LYNX_1	LynX-1 CE switch
VTSS_TARGET E_STAX_III_48	E-StaX-III-48
VTSS_TARGET E_STAX_III_68	E-StaX-III-68
VTSS_TARGET E_STAX_III_24_DUAL	Dual E-StaX-III-24

Enumerator

VTSS_TARGET_E_STAX_III_68_DUAL	Dual E-StaX-III-68
VTSS_TARGET_DAYTONA	Daytona FEC OTN Phy
VTSS_TARGET_TALLADEGA	Talladega FEC OTN Phy
VTSS_TARGET_SERVAL_2	Serval-2 CE switch
VTSS_TARGET_LYNX_2	LynX-2 CE switch
VTSS_TARGET_JAGUAR_2	Jaguar-2 CE switch
VTSS_TARGET_SPARX_IV_52	Sparx-IV-52 switch
VTSS_TARGET_SPARX_IV_44	Sparx-IV-44 switch
VTSS_TARGET_SPARX_IV_80	Sparx-IV-80 switch
VTSS_TARGET_SPARX_IV_90	Sparx-IV-80 switch

Definition at line 42 of file vtss_init_api.h.

8.15.4.2 vtss_port_mux_mode_t

```
enum vtss_port_mux_mode_t
```

Port mux configuration.

Enumerator

VTSS_PORT_MUX_MODE_0	Ports muxed to Serdes blocks: 24xSGMII, 4x10Gb, 1xRGMII
VTSS_PORT_MUX_MODE_1	Ports muxed to Serdes blocks: 20xSGMII, 4x2G5, 3x10G, 1xRGMII
VTSS_PORT_MUX_MODE_7	Ports muxed to Serdes blocks: 16xSGMII, 8x2G5, 2x10G, 1xRGMII

Definition at line 335 of file vtss_init_api.h.

8.15.4.3 vtss_qs_mode_t

```
enum vtss_qs_mode_t
```

The major modes of QS.

Enumerator

VTSS_QS_MODE_DISABLED	Manual settings are disabled, defaults are used
VTSS_QS_MODE_ENABLED	Manual settings are enabled.

Definition at line 390 of file vtss_init_api.h.

8.15.4.4 vtss_restart_t

enum `vtss_restart_t`

Restart type.

Enumerator

VTSS_RESTART_COLD	Cold: Chip and CPU restart, e.g. power cycling
VTSS_RESTART_COOL	Cool: Chip and CPU restart done by CPU
VTSS_RESTART_WARM	Warm: CPU restart only

Definition at line 601 of file vtss_init_api.h.

8.15.5 Function Documentation

8.15.5.1 vtss_inst_get()

```
vtss_rc vtss_inst_get (
    const vtss_target_type_t target,
    vtss_inst_create_t *const create )
```

Initialize create structure for target.

Parameters

<i>target</i>	[IN] Target name
<i>create</i>	[IN] Create structure

Returns

Return code.

8.15.5.2 vtss_inst_create()

```
vtss_rc vtss_inst_create (
    const vtss_inst_create_t *const create,
    vtss_inst_t *const inst )
```

Create target instance.

Parameters

<i>create</i>	[IN] Create structure
<i>inst</i>	[OUT] Target instance reference.

Returns

Return code.

8.15.5.3 vtss_inst_destroy()

```
vtss_rc vtss_inst_destroy (
    const vtss_inst_t inst )
```

Destroy target instance.

Parameters

<i>inst</i>	[IN] Target instance reference.
-------------	---------------------------------

Returns

Return code.

8.15.5.4 vtss_init_conf_get()

```
vtss_rc vtss_init_conf_get (
    const vtss_inst_t inst,
    vtss_init_conf_t *const conf )
```

Get default initialization configuration.

Parameters

<i>inst</i>	[IN] Target instance reference
<i>conf</i>	[OUT] Initialization configuration

Returns

Return code.

8.15.5.5 vtss_init_conf_set()

```
vtss_rc vtss_init_conf_set (
    const vtss_inst_t inst,
    const vtss_init_conf_t *const conf )
```

Set initialization configuration.

Parameters

<i>inst</i>	[IN] Target instance reference
<i>conf</i>	[IN] Initialization configuration

Returns

Return code.

8.15.5.6 vtss_restart_conf_end()

```
vtss_rc vtss_restart_conf_end (
    const vtss_inst_t inst )
```

Indicate configuration end. If a warm start has been done, the stored configuration will be applied.

Parameters

<i>inst</i>	[IN] Target instance reference
-------------	--------------------------------

Returns

Return code.

8.15.5.7 vtss_restart_status_get()

```
vtss_rc vtss_restart_status_get (
    const vtss_inst_t inst,
    vtss_restart_status_t *const status )
```

Get restart status.

Parameters

<i>inst</i>	[IN] Target instance reference
<i>status</i>	[OUT] Restart status

Returns

Return code.

8.15.5.8 vtss_restart_conf_get()

```
vtss_rc vtss_restart_conf_get (
    const vtss_inst_t inst,
    const vtss_restart_t *const restart )
```

Get restart configuration (next restart mode)

Parameters

<i>inst</i>	[IN] Target instance reference
<i>restart</i>	[OUT] Restart mode

Returns

Return code.

8.15.5.9 vtss_restart_conf_set()

```
vtss_rc vtss_restart_conf_set (
    const vtss_inst_t inst,
    const vtss_restart_t restart )
```

Set restart configuration (next restart mode)

Parameters

<i>inst</i>	[IN] Target instance reference
<i>restart</i>	[IN] Restart mode

Returns

Return code.

8.15.5.10 vtss_qs_conf_set()

```
vtss_rc vtss_qs_conf_set (
    const vtss_inst_t inst,
    const vtss_qs_conf_t *const qs )
```

Configure the Queue System.

Parameters

<i>inst</i>	[IN] Target instance reference
<i>qs</i>	[IN] The configuration of the queue system

Returns

Return code.

8.15.5.11 vtss_qs_conf_get()

```
vtss_rc vtss_qs_conf_get (
    const vtss_inst_t inst,
    vtss_qs_conf_t *const qs )
```

Get the configuration of the Queue System.

Parameters

<i>inst</i>	[IN] Target instance reference
<i>qs</i>	[OUT] The configuration of the queue system

Returns

Return code.

8.16 vtss_api/include/vtss_l2_api.h File Reference

Layer 2 API.

```
#include <vtss/api/types.h>
#include <vtss_security_api.h>
#include "vtss_port_api.h"
#include "vtss_packet_api.h"
#include <vtss/api/l2_types.h>
```

Data Structures

- struct [vtss_mac_table_entry_t](#)
MAC address entry.
- struct [vtss_mac_table_status_t](#)
MAC address table status.
- struct [vtss_learn_mode_t](#)
Learning mode.

- struct [vtss_vlan_conf_t](#)
VLAN configuration.
- struct [vtss_vlan_port_conf_t](#)
VLAN port configuration.
- struct [vtss_vlan_vid_conf_t](#)
VLAN ID configuration.
- struct [vtss_vcl_port_conf_t](#)
VCL port configuration.
- struct [vtss_vce_mac_t](#)
VCE MAC header information.
- struct [vtss_vce_tag_t](#)
VCE tag information.
- struct [vtss_vce_frame_etype_t](#)
Frame data for VTSS_VCE_TYPE_ETYPE.
- struct [vtss_vce_frame_llc_t](#)
Frame data for VTSS_VCE_TYPE_LLCC.
- struct [vtss_vce_frame_snap_t](#)
Frame data for VTSS_VCE_TYPE_SNAP.
- struct [vtss_vce_frame_ipv4_t](#)
Frame data for VTSS_VCE_TYPE_IPV4.
- struct [vtss_vce_frame_ipv6_t](#)
Frame data for VTSS_VCE_TYPE_IPV6.
- struct [vtss_vce_key_t](#)
VCE Key.
- struct [vtss_vce_action_t](#)
VCE Action.
- struct [vtss_vce_t](#)
VLAN Control Entry.
- struct [vtss_vlan_trans_port2grp_conf_t](#)
VLAN translation port-to-group configuration.
- struct [vtss_vlan_trans_grp2vlan_conf_t](#)
VLAN translation group-to-VLAN configuration.
- struct [vtss_dgroup_port_conf_t](#)
Destination group port configuration.
- struct [vtss_sflow_port_conf_t](#)
sFlow configuration structure.
- struct [vtss_vstax_glag_entry_t](#)
GLAG info.
- struct [vtss_mirror_conf_t](#)
Mirror configuration.
- struct [vtss_eps_port_conf_t](#)
Port protection configuration.
- struct [vtss_vstax_conf_t](#)
VStaX configuration for switch.
- struct [vtss_vstax_port_conf_t](#)
VStaX setup for port.
- struct [vtss_vstax_route_entry_t](#)
UPSID Route Entry.
- struct [vtss_vstax_route_table_t](#)
UPSID Route Table.

Macros

- #define VTSS_MAC_ADDRS 32768
- #define VTSS_VSTAX_UPSIDS (32)
- #define VTSS_VSTAX_UPSID_START (0)
- #define VTSS_VSTAX_UPSID_MIN VTSS_VSTAX_UPSID_START
- #define VTSS_VSTAX_UPSID_MAX (VTSS_VSTAX_UPSID_MIN+VTSS_VSTAX_UPSIDS - 1)
- #define VTSS_VSTAX_UPSID_LEGAL(upsid) (VTSS_VSTAX_UPSID_MIN <= (upsid) && (upsid) <= VTSS_VSTAX_UPSID_MAX)
- #define VTSS_VSTAX_UPSID_UNDEF (-1)
- #define VTSS_UPSPN_CPU 0xffffffff
- #define VTSS_UPSPN_NONE 0xffffffff
- #define VTSS_MSTIS (65)
- #define VTSS_MSTI_START (0)
- #define VTSS_MSTI_END (VTSS_MSTI_START+VTSS_MSTIS)
- #define VTSS_MSTI_ARRAY_SIZE VTSS_MSTI_END
- #define VTSS_VCL_IDS 256
- #define VTSS_VCL_ID_START 0
- #define VTSS_VCL_ID_END (VTSS_VCL_ID_START+VTSS_VCL_IDS)
- #define VTSS_VCL_ARRAY_SIZE VTSS_VCL_ID_END
- #define VTSS_VCE_ID_LAST 0
- #define VTSS_VLAN_TRANS_GROUP_MAX_CNT VTSS_PORTS
- #define VTSS_VLAN_TRANS_MAX_CNT 256
- #define VTSS_VLAN_TRANS_NULL_GROUP_ID 0
- #define VTSS_VLAN_TRANS_FIRST_GROUP_ID 1
- #define VTSS_VLAN_TRANS_VID_START 1
- #define VTSS_VLAN_TRANS_MAX_VLAN_ID 4095
- #define VTSS_VLAN_TRANS_LAST_GROUP_ID (VTSS_VLAN_TRANS_FIRST_GROUP_ID + VTSS_VLAN_TRANS_GROUP - 1)
- #define VTSS_VLAN_TRANS_VALID_GROUP_CHECK(grp_id)
- #define VTSS_VLAN_TRANS_VALID_VLAN_CHECK(vid)
- #define VTSS_VLAN_TRANS_NULL_CHECK(ptr) ((ptr == NULL) ? FALSE : TRUE)
- #define VTSS_VLAN_TRANS_PORT_BF_SIZE ((VTSS_PORTS + 7)/8)
- #define VTSS_PVLANS (VTSS_PORTS)
- #define VTSS_PVLAN_NO_START (0)
- #define VTSS_PVLAN_NO_END (VTSS_PVLAN_NO_START+VTSS_PVLANS)
- #define VTSS_PVLAN_ARRAY_SIZE VTSS_PVLAN_NO_END
- #define VTSS_PVLAN_NO_DEFAULT (0)
- #define VTSS_ERPIS (64)
- #define VTSS_ERPI_START (0)
- #define VTSS_ERPI_END (VTSS_ERPI_START+VTSS_ERPIS)
- #define VTSS_ERPI_ARRAY_SIZE VTSS_ERPI_END

Typedefs

- typedef int **vtss_vstax_upsid_t**
VStaX Unit Port Set ID (UPSID; 0-31).
- typedef u32 **vtss_vstax_upspn_t**
Unit Port Set Port Number.
- typedef u32 **vtss_mac_table_age_time_t**
MAC address table age time.
- typedef u32 **vtss_msti_t**
MSTP instance number.

- `typedef u32 vtss_vce_id_t`
VCE ID type.
- `typedef u64 vtss_vt_id_t`
- `typedef u32 vtss_pvlan_no_t`
Private VLAN Number.
- `typedef vtss_port_no_t vtss_dgroup_no_t`
EVC policer configuration.
- `typedef u32 vtss_erpi_t`
ERPS instance number.

Enumerations

- `enum vtss_stp_state_t { VTSS_STP_STATE_DISCARDING, VTSS_STP_STATE_LEARNING, VTSS_STP_STATE_FORWARDING }`
Spanning Tree state.
- `enum vtss_vlan_port_type_t { VTSS_VLAN_PORT_TYPE_UNAWARE, VTSS_VLAN_PORT_TYPE_C, VTSS_VLAN_PORT_TYPE_S, VTSS_VLAN_PORT_TYPE_S_CUSTOM }`
VLAN port type configuration.
- `enum vtss_vlan_tx_tag_t { VTSS_VLAN_TX_TAG_PORT, VTSS_VLAN_TX_TAG_DISABLE, VTSS_VLAN_TX_TAG_ENABLE }`
VLAN Tx tag type.
- `enum vtss_vce_type_t { VTSS_VCE_TYPE_ANY, VTSS_VCE_TYPE_ETYPE, VTSS_VCE_TYPE_LLC, VTSS_VCE_TYPE_SNAP, VTSS_VCE_TYPE_IPV4, VTSS_VCE_TYPE_IPV6 }`
VCE frame type.
- `enum vtss_mirror_tag_t { VTSS_MIRROR_TAG_NONE, VTSS_MIRROR_TAG_C, VTSS_MIRROR_TAG_S, VTSS_MIRROR_TAG_S_CUSTOM }`
Mirror port configuration.
- `enum vtss_eps_port_type_t { VTSS_EPS_PORT_1_PLUS_1, VTSS_EPS_PORT_1_FOR_1 }`
Port protection type.
- `enum vtss_eps_selector_t { VTSS_EPS_SELECTOR_WORKING, VTSS_EPS_SELECTOR_PROTECTION }`
EPS selector.
- `enum vtss_erps_state_t { VTSS_ERPS_STATE_FORWARDING, VTSS_ERPS_STATE_DISCARDING }`
ERPS state.
- `enum vtss_vstax_topology_type_t { VTSS_VSTAX_TOPOLOGY_CHAIN, VTSS_VSTAX_TOPOLOGY_RING }`
VStaX topology type.

Functions

- `vtss_rc vtss_mac_table_add (const vtss_inst_t inst, const vtss_mac_table_entry_t *const entry)`
Add MAC address entry.
- `vtss_rc vtss_mac_table_del (const vtss_inst_t inst, const vtss_vid_mac_t *const vid_mac)`
Delete MAC address entry.
- `vtss_rc vtss_mac_table_get (const vtss_inst_t inst, const vtss_vid_mac_t *const vid_mac, vtss_mac_table_entry_t *const entry)`
Get MAC address entry.
- `vtss_rc vtss_mac_table_get_next (const vtss_inst_t inst, const vtss_vid_mac_t *const vid_mac, vtss_mac_table_entry_t *const entry)`
Lookup next MAC address entry.

- `vtss_rc vtss_mac_table_age_time_get (const vtss_inst_t inst, vtss_mac_table_age_time_t *const age_time)`
Get MAC address table age time.
- `vtss_rc vtss_mac_table_age_time_set (const vtss_inst_t inst, const vtss_mac_table_age_time_t age_time)`
Set MAC address table age time.
- `vtss_rc vtss_mac_table_age (const vtss_inst_t inst)`
Do age scan of the MAC address table. This should be done periodically with interval T/2, where T is the age timer.
- `vtss_rc vtss_mac_table_vlan_age (const vtss_inst_t inst, const vtss_vid_t vid)`
Do VLAN specific age scan of the MAC address table.
- `vtss_rc vtss_mac_table_flush (const vtss_inst_t inst)`
Flush MAC address table, i.e. remove all unlocked entries.
- `vtss_rc vtss_mac_table_port_flush (const vtss_inst_t inst, const vtss_port_no_t port_no)`
Delete MAC address entries learned on port.
- `vtss_rc vtss_mac_table_vlan_flush (const vtss_inst_t inst, const vtss_vid_t vid)`
Delete MAC address entries learned on VLAN ID.
- `vtss_rc vtss_mac_table_vlan_port_flush (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_vid_t vid)`
Delete MAC address entries learned on port and VLAN ID.
- `vtss_rc vtss_mac_table_upsid_flush (const vtss_inst_t inst, const vtss_vstax_upsid_t upsid)`
Delete MAC address entries learned on UPSID.
- `vtss_rc vtss_mac_table_upsid_upspn_flush (const vtss_inst_t inst, const vtss_vstax_upsid_t upsid, const vtss_vstax_upspn_t upspn)`
Delete MAC address entries learned on (UPSID, UPSPN).
- `vtss_rc vtss_mac_table_glag_add (const vtss_inst_t inst, const vtss_mac_table_entry_t *const entry, const vtss_glag_no_t glag_no)`
Learn MAC address entry on GLAG.
- `vtss_rc vtss_mac_table_glag_flush (const vtss_inst_t inst, const vtss_glag_no_t glag_no)`
Delete MAC address entries learned on GLAG.
- `vtss_rc vtss_mac_table_vlan_glag_flush (const vtss_inst_t inst, const vtss_glag_no_t glag_no, const vtss_vid_t vid)`
Delete MAC address entries learned on GLAG and VID.
- `vtss_rc vtss_mac_table_status_get (const vtss_inst_t inst, vtss_mac_table_status_t *const status)`
Get MAC address table status.
- `vtss_rc vtss_learn_port_mode_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_learn_mode_t *const mode)`
Get the learn mode for a port.
- `vtss_rc vtss_learn_port_mode_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_learn_mode_t *const mode)`
Set the learn mode for a port.
- `vtss_rc vtss_port_state_get (const vtss_inst_t inst, const vtss_port_no_t port_no, BOOL *const state)`
Get port operational state.
- `vtss_rc vtss_port_state_set (const vtss_inst_t inst, const vtss_port_no_t port_no, BOOL state)`
Set port operational state.
- `vtss_rc vtss_stp_port_state_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_stp_state_t *const state)`
Get Spanning Tree state for a port.
- `vtss_rc vtss_stp_port_state_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_stp_state_t state)`
Set Spanning Tree state for a port.
- `vtss_rc vtss_mstp_vlan_msti_get (const vtss_inst_t inst, const vtss_vid_t vid, vtss_msti_t *const msti)`
Get MSTP instance mapping for a VLAN.
- `vtss_rc vtss_mstp_vlan_msti_set (const vtss_inst_t inst, const vtss_vid_t vid, const vtss_msti_t msti)`
Set MSTP instance mapping for a VLAN.

- `vtss_rc vtss_mstp_port_msti_state_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_msti_t` msti, `vtss_stp_state_t` *const state)

Get MSTP state for a port and MSTP instance.
- `vtss_rc vtss_mstp_port_msti_state_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_msti_t` msti, const `vtss_stp_state_t` state)

Set MSTP state for a port and MSTP instance.
- `vtss_rc vtss_vlan_conf_get` (const `vtss_inst_t` inst, `vtss_vlan_conf_t` *const conf)

Get VLAN configuration.
- `vtss_rc vtss_vlan_conf_set` (const `vtss_inst_t` inst, const `vtss_vlan_conf_t` *const conf)

Set VLAN configuration.
- `vtss_rc vtss_vlan_port_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_vlan_port_conf_t` *const conf)

Get VLAN mode for port.
- `vtss_rc vtss_vlan_port_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_vlan_port_conf_t` *const conf)

Set VLAN mode for port.
- `vtss_rc vtss_vlan_port_members_get` (const `vtss_inst_t` inst, const `vtss_vid_t` vid, `BOOL` member[`VTSS_PORT_ARRAY_SIZE`])

Get VLAN membership.
- `vtss_rc vtss_vlan_port_members_set` (const `vtss_inst_t` inst, const `vtss_vid_t` vid, const `BOOL` member[`VTSS_PORT_ARRAY_SIZE`])

Set VLAN membership.
- `vtss_rc vtss_vlan_vid_conf_get` (const `vtss_inst_t` inst, const `vtss_vid_t` vid, `vtss_vlan_vid_conf_t` *const conf)

Get VLAN ID configuration.
- `vtss_rc vtss_vlan_vid_conf_set` (const `vtss_inst_t` inst, const `vtss_vid_t` vid, const `vtss_vlan_vid_conf_t` *const conf)

Set VLAN ID configuration.
- `vtss_rc vtss_vlan_tx_tag_get` (const `vtss_inst_t` inst, const `vtss_vid_t` vid, `vtss_vlan_tx_tag_t` tx←tag[`VTSS_PORT_ARRAY_SIZE`])

Get VLAN Tx tagging configuration.
- `vtss_rc vtss_vlan_tx_tag_set` (const `vtss_inst_t` inst, const `vtss_vid_t` vid, const `vtss_vlan_tx_tag_t` tx←tag[`VTSS_PORT_ARRAY_SIZE`])

Get VLAN Tx tagging configuration.
- `vtss_rc vtss_vcl_port_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_vcl_port_conf_t` *const conf)

Get VCL port configuration.
- `vtss_rc vtss_vcl_port_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_vcl_port_conf_t` *const conf)

Get VCL port configuration.
- `vtss_rc vtss_vce_init` (const `vtss_inst_t` inst, const `vtss_vce_type_t` type, `vtss_vce_t` *const vce)

Initialize VCE to default values.
- `vtss_rc vtss_vce_add` (const `vtss_inst_t` inst, const `vtss_vce_id_t` vce_id, const `vtss_vce_t` *const vce)

Add/modify VCE.
- `vtss_rc vtss_vce_del` (const `vtss_inst_t` inst, const `vtss_vce_id_t` vce_id)

Delete VCE.
- `vtss_rc vtss_vlan_trans_group_add` (const `vtss_inst_t` inst, const `u16` group_id, const `vtss_vid_t` vid, const `vtss_vid_t` trans_vid)

Create VLAN Translation Group entry.
- `vtss_rc vtss_vlan_trans_group_del` (const `vtss_inst_t` inst, const `u16` group_id, const `vtss_vid_t` vid)

Delete VLAN Translation Group entry.
- `vtss_rc vtss_vlan_trans_group_get` (const `vtss_inst_t` inst, `vtss_vlan_trans_grp2vlan_conf_t` *conf, `BOOL` next)

Get VLAN Translation Group entry.

- `vtss_rc vtss_vlan_trans_group_to_port_set` (const `vtss_inst_t` inst, const `vtss_vlan_trans_port2grp_conf_t` *conf)
 - Get VLAN Translation Group entry.
 - Associate VLAN Translation Group entries to a port_list. Only one port can be part of one group not multiple groups.
- `vtss_rc vtss_vlan_trans_group_to_port_get` (const `vtss_inst_t` inst, `vtss_vlan_trans_port2grp_conf_t` *conf, `BOOL` next)
 - VLAN Translation function to fetch all ports for a group.
- `vtss_rc vtss_isolated_vlan_get` (const `vtss_inst_t` inst, const `vtss_vid_t` vid, `BOOL` *const isolated)
 - Get enable/disable port isolation for VLAN.
- `vtss_rc vtss_isolated_vlan_set` (const `vtss_inst_t` inst, const `vtss_vid_t` vid, const `BOOL` isolated)
 - Set enable/disable port isolation for VLAN.
- `vtss_rc vtss_isolated_port_members_get` (const `vtss_inst_t` inst, `BOOL` member[`VTSS_PORT_ARRAY_SIZE`])
 - Get the isolated port member set.
- `vtss_rc vtss_isolated_port_members_set` (const `vtss_inst_t` inst, const `BOOL` member[`VTSS_PORT_ARRAY_SIZE`])
 - Set the isolated port member set.
- `vtss_rc vtss_pvlan_port_members_get` (const `vtss_inst_t` inst, const `vtss_pvlan_no_t` pvlan_no, `BOOL` member[`VTSS_PORT_ARRAY_SIZE`])
 - Get Private VLAN membership.
- `vtss_rc vtss_pvlan_port_members_set` (const `vtss_inst_t` inst, const `vtss_pvlan_no_t` pvlan_no, const `BOOL` member[`VTSS_PORT_ARRAY_SIZE`])
 - Set Private VLAN membership.
- `vtss_rc vtss_apvlan_port_members_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `BOOL` member[`VTSS_PORT_ARRAY_SIZE`])
 - Get Asymmetric Private VLAN membership.
- `vtss_rc vtss_apvlan_port_members_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `BOOL` member[`VTSS_PORT_ARRAY_SIZE`])
 - Set Asymmetric Private VLAN membership.
- `vtss_rc vtss_dgroup_port_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_dgroup_port_conf_t` *const conf)
 - Get Destination Group configuration for port.
- `vtss_rc vtss_dgroup_port_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_dgroup_port_conf_t` *const conf)
 - Set Destination Group configuration for port.
- `vtss_rc vtss_sflow_port_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_sflow_port_conf_t` *const conf)
 - Get port sFlow configuration.
- `vtss_rc vtss_sflow_port_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_sflow_port_conf_t` *const conf)
 - Set port sFlow configuration.
- `vtss_rc vtss_sflow_sampling_rate_convert` (const `vtss_inst_t` inst, const `BOOL` power2, const `u32` rate_in, `u32` *const rate_out)
 - Convert desired sample rate to supported sample rate.
- `vtss_rc vtss_aggr_port_members_get` (const `vtss_inst_t` inst, const `vtss_aggr_no_t` aggr_no, `BOOL` member[`VTSS_PORT_ARRAY_SIZE`])
 - Get aggregation port members.
- `vtss_rc vtss_aggr_port_members_set` (const `vtss_inst_t` inst, const `vtss_aggr_no_t` aggr_no, const `BOOL` member[`VTSS_PORT_ARRAY_SIZE`])
 - Set aggregation port members.
- `vtss_rc vtss_aggr_mode_get` (const `vtss_inst_t` inst, `vtss_aggr_mode_t` *const mode)
 - Get aggregation traffic distribution mode.
- `vtss_rc vtss_aggr_mode_set` (const `vtss_inst_t` inst, const `vtss_aggr_mode_t` *const mode)
 - Set aggregation traffic distribution mode.

- `vtss_rc vtss_aggr_glag_members_get` (const `vtss_inst_t` inst, const `vtss_glag_no_t` glag_no, `BOOL` member[`VTSS_PORT_ARRAY_SIZE`])
Get global aggregation port members.
- `vtss_rc vtss_vstax_glag_get` (const `vtss_inst_t` inst, const `vtss_glag_no_t` glag_no, `vtss_vstax_glag_entry_t` entry[`VTSS_GLAG_PORT_ARRAY_SIZE`])
Get global aggregation port members.
- `vtss_rc vtss_vstax_glag_set` (const `vtss_inst_t` inst, const `vtss_glag_no_t` glag_no, const `vtss_vstax_glag_entry_t` entry[`VTSS_GLAG_PORT_ARRAY_SIZE`])
Get global aggregation port members.
- `vtss_rc vtss_mirror_conf_get` (const `vtss_inst_t` inst, `vtss_mirror_conf_t` *const conf)
Get the mirror configuration.
- `vtss_rc vtss_mirror_conf_set` (const `vtss_inst_t` inst, const `vtss_mirror_conf_t` *const conf)
Set the mirror configuration.
- `vtss_rc vtss_mirror_monitor_port_get` (const `vtss_inst_t` inst, `vtss_port_no_t` *const port_no)
Get the mirror monitor port.
- `vtss_rc vtss_mirror_monitor_port_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no)
Set the mirror monitor port.
- `vtss_rc vtss_mirror_ingress_ports_get` (const `vtss_inst_t` inst, `BOOL` member[`VTSS_PORT_ARRAY_SIZE`])
Get the mirror ingress ports.
- `vtss_rc vtss_mirror_ingress_ports_set` (const `vtss_inst_t` inst, const `BOOL` member[`VTSS_PORT_ARRAY_SIZE`])
Set the mirror ingress ports.
- `vtss_rc vtss_mirror_egress_ports_get` (const `vtss_inst_t` inst, `BOOL` member[`VTSS_PORT_ARRAY_SIZE`])
Get the mirror egress ports.
- `vtss_rc vtss_mirror_egress_ports_set` (const `vtss_inst_t` inst, const `BOOL` member[`VTSS_PORT_ARRAY_SIZE`])
Set the mirror egress ports.
- `vtss_rc vtss_mirror_cpu_ingress_get` (const `vtss_inst_t` inst, `BOOL` *member)
Get the mirror CPU ingress.
- `vtss_rc vtss_mirror_cpu_ingress_set` (const `vtss_inst_t` inst, const `BOOL` member)
Set CPU ingress mirroring.
- `vtss_rc vtss_mirror_cpu_egress_get` (const `vtss_inst_t` inst, `BOOL` *member)
Get the mirror CPU egress.
- `vtss_rc vtss_mirror_cpu_egress_set` (const `vtss_inst_t` inst, const `BOOL` member)
Set the mirror CPU egress.
- `vtss_rc vtss_uc_flood_members_get` (const `vtss_inst_t` inst, `BOOL` member[`VTSS_PORT_ARRAY_SIZE`])
Get unicast flood members.
- `vtss_rc vtss_uc_flood_members_set` (const `vtss_inst_t` inst, const `BOOL` member[`VTSS_PORT_ARRAY_SIZE`])
Set unicast flood members.
- `vtss_rc vtss_mc_flood_members_get` (const `vtss_inst_t` inst, `BOOL` member[`VTSS_PORT_ARRAY_SIZE`])
Get multicast flood members.
- `vtss_rc vtss_mc_flood_members_set` (const `vtss_inst_t` inst, const `BOOL` member[`VTSS_PORT_ARRAY_SIZE`])
Set multicast flood members.
- `vtss_rc vtss_ipv4_mc_flood_members_get` (const `vtss_inst_t` inst, `BOOL` member[`VTSS_PORT_ARRAY_SIZE`])
Get IPv4 multicast flood members.
- `vtss_rc vtss_ipv4_mc_flood_members_set` (const `vtss_inst_t` inst, const `BOOL` member[`VTSS_PORT_ARRAY_SIZE`])
Set IPv4 multicast flood members.
- `vtss_rc vtss_ipv6_mc_flood_members_get` (const `vtss_inst_t` inst, `BOOL` member[`VTSS_PORT_ARRAY_SIZE`])
Get IPv6 multicast flood members.
- `vtss_rc vtss_ipv6_mc_flood_members_set` (const `vtss_inst_t` inst, const `BOOL` member[`VTSS_PORT_ARRAY_SIZE`])
Set IPv6 multicast flood members.
- `vtss_rc vtss_ipv6_mc_ctrl_flood_get` (const `vtss_inst_t` inst, `BOOL` *const scope)
Get IPv6 multicast control flooding mode.

- `vtss_rc vtss_ipv6_mc_ctrl_flood_set` (const `vtss_inst_t` inst, const `BOOL` scope)

Set IPv6 multicast control flooding mode. This controls whether unknown Link-Local scope IPv6 multicasts (FF02::/16) are flooded to all ports or to the ports in the IPv6 multicast flood mask.
- `vtss_rc vtss_eps_port_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_eps_port_conf_t` *const conf)

Get EPS port configuration.
- `vtss_rc vtss_eps_port_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_eps_port_conf_t` *const conf)

Set EPS port configuration.
- `vtss_rc vtss_eps_port_selector_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_eps_selector_t` *const selector)

Get EPS port selector.
- `vtss_rc vtss_eps_port_selector_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_eps_selector_t` selector)

Set EPS port selector.
- `vtss_rc vtss_erps_vlan_member_get` (const `vtss_inst_t` inst, const `vtss_erpi_t` erpi, const `vtss_vid_t` vid, `BOOL` *const member)

Get ERPS member state for a VLAN.
- `vtss_rc vtss_erps_vlan_member_set` (const `vtss_inst_t` inst, const `vtss_erpi_t` erpi, const `vtss_vid_t` vid, const `BOOL` member)

Set ERPS member state for a VLAN.
- `vtss_rc vtss_erps_port_state_get` (const `vtss_inst_t` inst, const `vtss_erpi_t` erpi, const `vtss_port_no_t` port_no, `vtss_erps_state_t` *const state)

Get ERPS state for ERPS instance and port.
- `vtss_rc vtss_erps_port_state_set` (const `vtss_inst_t` inst, const `vtss_erpi_t` erpi, const `vtss_port_no_t` port_no, const `vtss_erps_state_t` state)

Set ERPS state for ERPS instance and port.
- `vtss_rc vtss_vstax_conf_get` (const `vtss_inst_t` inst, `vtss_vstax_conf_t` *const conf)

Get VStaX configuration for switch.
- `vtss_rc vtss_vstax_conf_set` (const `vtss_inst_t` inst, const `vtss_vstax_conf_t` *const conf)

Set VStaX configuration for switch.
- `vtss_rc vtss_vstax_port_conf_get` (const `vtss_inst_t` inst, const `vtss_chip_no_t` chip_no, const `BOOL` stack_port_a, `vtss_vstax_port_conf_t` *const conf)

Get VStaX configuration for port.
- `vtss_rc vtss_vstax_port_conf_set` (const `vtss_inst_t` inst, const `vtss_chip_no_t` chip_no, const `BOOL` stack_port_a, const `vtss_vstax_port_conf_t` *const conf)

Set VStaX configuration for port.
- `vtss_rc vtss_vstax_master_upsid_get` (const `vtss_inst_t` inst, `vtss_vstax_upsid_t` *const master_upsid)

Get UPSID of current master in stack.
- `vtss_rc vtss_vstax_master_upsid_set` (const `vtss_inst_t` inst, const `vtss_vstax_upsid_t` master_upsid)

Set UPSID of current master in stack.
- `vtss_rc vtss_vstax_topology_set` (const `vtss_inst_t` inst, const `vtss_chip_no_t` chip_no, const `vtss_vstax_route_table_t` *table)

Set stack topology.

8.16.1 Detailed Description

Layer 2 API.

This header file describes Layer 2 switching functions

8.16.2 Macro Definition Documentation

8.16.2.1 VTSS_MAC_ADDRS

```
#define VTSS_MAC_ADDRS 32768
```

Number of MAC addresses

Definition at line 93 of file vtss_l2_api.h.

8.16.2.2 VTSS_VSTAX_UPSIDS

```
#define VTSS_VSTAX_UPSIDS (32)
```

Number of UPSIDs

Definition at line 102 of file vtss_l2_api.h.

8.16.2.3 VTSS_VSTAX_UPSID_START

```
#define VTSS_VSTAX_UPSID_START ( 0 )
```

First UPSID value

Definition at line 103 of file vtss_l2_api.h.

8.16.2.4 VTSS_VSTAX_UPSID_MIN

```
#define VTSS_VSTAX_UPSID_MIN VTSS_VSTAX_UPSID_START
```

Minimum UPSID value

Definition at line 104 of file vtss_l2_api.h.

8.16.2.5 VTSS_VSTAX_UPSID_MAX

```
#define VTSS_VSTAX_UPSID_MAX (VTSS_VSTAX_UPSID_MIN+VTSS_VSTAX_UPSIDS - 1)
```

Maximum UPSID value

Definition at line 105 of file vtss_l2_api.h.

8.16.2.6 VTSS_VSTAX_UPSID_LEGAL

```
#define VTSS_VSTAX_UPSID_LEGAL(upsid) (VTSS_VSTAX_UPSID_MIN <= (upsid) && (upsid) <= VTSS_VSTAX_UPSID_MAX)
```

Checks if UPSIDs is legal

Definition at line 106 of file vtss_l2_api.h.

8.16.2.7 VTSS_VSTAX_UPSID_UNDEF

```
#define VTSS_VSTAX_UPSID_UNDEF (-1)
```

Undefined UPSID. Only applicable in selected contexts

Definition at line 107 of file vtss_l2_api.h.

8.16.2.8 VTSS_UPSPN_CPU

```
#define VTSS_UPSPN_CPU 0xffffffff
```

MAC address entry is from CPU

Definition at line 114 of file vtss_l2_api.h.

8.16.2.9 VTSS_UPSPN_NONE

```
#define VTSS_UPSPN_NONE 0xffffffff
```

Used to indicate end of GLAG list

Definition at line 115 of file vtss_l2_api.h.

8.16.2.10 VTSS_MSTIS

```
#define VTSS_MSTIS (65)
```

Number of MSTP instances

Definition at line 514 of file vtss_l2_api.h.

8.16.2.11 VTSS_MSTI_START

```
#define VTSS_MSTI_START (0)
```

MSTI start number

Definition at line 515 of file vtss_l2_api.h.

8.16.2.12 VTSS_MSTI_END

```
#define VTSS_MSTI_END (VTSS_MSTI_START+VTSS_MSTIS)
```

MSTI end number

Definition at line 516 of file vtss_l2_api.h.

8.16.2.13 VTSS_MSTI_ARRAY_SIZE

```
#define VTSS_MSTI_ARRAY_SIZE VTSS_MSTI_END
```

MSTI array size

Definition at line 517 of file vtss_l2_api.h.

8.16.2.14 VTSS_VCL_IDS

```
#define VTSS_VCL_IDS 256
```

Number of VCLs

Definition at line 922 of file vtss_l2_api.h.

8.16.2.15 VTSS_VCL_ID_START

```
#define VTSS_VCL_ID_START 0
```

VCL ID start number

Definition at line 923 of file vtss_l2_api.h.

8.16.2.16 VTSS_VCL_ID_END

```
#define VTSS_VCL_ID_END (VTSS_VCL_ID_START+VTSS_VCL_IDS)
```

VCL ID end number

Definition at line 924 of file vtss_l2_api.h.

8.16.2.17 VTSS_VCL_ARRAY_SIZE

```
#define VTSS_VCL_ARRAY_SIZE VTSS_VCL_ID_END
```

VCL ID array size

Definition at line 925 of file vtss_l2_api.h.

8.16.2.18 VTSS_VCE_ID_LAST

```
#define VTSS_VCE_ID_LAST 0
```

Special value used to add last in list

Definition at line 927 of file vtss_l2_api.h.

8.16.2.19 VTSS_VLAN_TRANS_GROUP_MAX_CNT

```
#define VTSS_VLAN_TRANS_GROUP_MAX_CNT VTSS_PORTS
```

Maximum VLAN Translation Groups count

Definition at line 1079 of file vtss_l2_api.h.

8.16.2.20 VTSS_VLAN_TRANS_MAX_CNT

```
#define VTSS_VLAN_TRANS_MAX_CNT 256
```

Maximum VLAN Translations per group count

Definition at line 1080 of file vtss_l2_api.h.

8.16.2.21 VTSS_VLAN_TRANS_NULL_GROUP_ID

```
#define VTSS_VLAN_TRANS_NULL_GROUP_ID 0
```

Special value for group ID

Definition at line 1081 of file vtss_l2_api.h.

8.16.2.22 VTSS_VLAN_TRANS_FIRST_GROUP_ID

```
#define VTSS_VLAN_TRANS_FIRST_GROUP_ID 1
```

First Group ID

Definition at line 1082 of file vtss_l2_api.h.

8.16.2.23 VTSS_VLAN_TRANS_VID_START

```
#define VTSS_VLAN_TRANS_VID_START 1
```

First valid VLAN ID

Definition at line 1083 of file vtss_l2_api.h.

8.16.2.24 VTSS_VLAN_TRANS_MAX_VLAN_ID

```
#define VTSS_VLAN_TRANS_MAX_VLAN_ID 4095
```

Last valid VLAN ID

Definition at line 1084 of file vtss_l2_api.h.

8.16.2.25 VTSS_VLAN_TRANS_LAST_GROUP_ID

```
#define VTSS_VLAN_TRANS_LAST_GROUP_ID (VTSS_VLAN_TRANS_FIRST_GROUP_ID + VTSS_VLAN_TRANS_GROUP_MAX_CNT - 1)
```

Last valid Group ID

Definition at line 1085 of file vtss_l2_api.h.

8.16.2.26 VTSS_VLAN_TRANS_VALID_GROUP_CHECK

```
#define VTSS_VLAN_TRANS_VALID_GROUP_CHECK( grp_id )
```

Value:

```
((grp_id < VTSS_VLAN_TRANS_FIRST_GROUP_ID) || \  
 (grp_id >  
 VTSS_VLAN_TRANS_LAST_GROUP_ID)) ? FALSE : TRUE)
```

Macro to check valid group

Definition at line 1087 of file vtss_l2_api.h.

8.16.2.27 VTSS_VLAN_TRANS_VALID_VLAN_CHECK

```
#define VTSS_VLAN_TRANS_VALID_VLAN_CHECK( vid )
```

Value:

```
((vid < VTSS_VLAN_TRANS_VID_START) || (vid > VTSS_VLAN_TRANS_MAX_VLAN_ID)) \  
 ? FALSE : TRUE)
```

Macro to check valid VLAN ID

Definition at line 1090 of file vtss_l2_api.h.

8.16.2.28 VTSS_VLAN_TRANS_NULL_CHECK

```
#define VTSS_VLAN_TRANS_NULL_CHECK( ptr ) ((ptr == NULL) ? FALSE : TRUE)
```

Macro to check NULL Pointer

Definition at line 1093 of file vtss_l2_api.h.

8.16.2.29 VTSS_VLAN_TRANS_PORT_BF_SIZE

```
#define VTSS_VLAN_TRANS_PORT_BF_SIZE ((VTSS_PORTS + 7)/8)
```

Macro Same as VTSS_PORT_BF_SIZE

Definition at line 1094 of file vtss_l2_api.h.

8.16.2.30 VTSS_PVLANS

```
#define VTSS_PVLANS (VTSS_PORTS)
```

Number of PVLANS

Definition at line 1304 of file vtss_l2_api.h.

8.16.2.31 VTSS_PVLAN_NO_START

```
#define VTSS_PVLAN_NO_START (0)
```

PVLAN start number

Definition at line 1305 of file vtss_l2_api.h.

8.16.2.32 VTSS_PVLAN_NO_END

```
#define VTSS_PVLAN_NO_END (VTSS_PVLAN_NO_START+VTSS_PVLANS)
```

PVLAN end number

Definition at line 1306 of file vtss_l2_api.h.

8.16.2.33 VTSS_PVLAN_ARRAY_SIZE

```
#define VTSS_PVLAN_ARRAY_SIZE VTSS_PVLAN_NO_END
```

PVLAN array size

Definition at line 1307 of file vtss_l2_api.h.

8.16.2.34 VTSS_PVLAN_NO_DEFAULT

```
#define VTSS_PVLAN_NO_DEFAULT (0)
```

Default PVLAN

Definition at line 1308 of file vtss_l2_api.h.

8.16.2.35 VTSS_ERPIS

```
#define VTSS_ERPIS (64)
```

Number of ERPS instances

Definition at line 2228 of file vtss_l2_api.h.

8.16.2.36 VTSS_ERPI_START

```
#define VTSS_ERPI_START (0)
```

ERPI start number

Definition at line 2229 of file vtss_l2_api.h.

8.16.2.37 VTSS_ERPI_END

```
#define VTSS_ERPI_END (VTSS_ERPI_START+VTSS_ERPIS)
```

ERPI end number

Definition at line 2230 of file vtss_l2_api.h.

8.16.2.38 VTSS_ERPI_ARRAY_SIZE

```
#define VTSS_ERPI_ARRAY_SIZE VTSS_ERPI_END
```

ERPI array size

Definition at line 2231 of file vtss_l2_api.h.

8.16.3 Typedef Documentation

8.16.3.1 vtss_vt_id_t

```
typedef u64 vtss_vt_id_t
```

VLAN Translation ID

Definition at line 1095 of file vtss_l2_api.h.

8.16.4 Enumeration Type Documentation

8.16.4.1 vtss_stp_state_t

```
enum vtss_stp_state_t
```

Spanning Tree state.

Enumerator

VTSS_STP_STATE_DISCARDING	STP state discarding (admin/operational down)
VTSS_STP_STATE_LEARNING	STP state learning
VTSS_STP_STATE_FORWARDING	STP state forwarding

Definition at line 474 of file vtss_l2_api.h.

8.16.4.2 vtss_vlan_port_type_t

enum [vtss_vlan_port_type_t](#)

VLAN port type configuration.

Enumerator

VTSS_VLAN_PORT_TYPE_UNAWARE	VLAN unaware port
VTSS_VLAN_PORT_TYPE_C	C-port
VTSS_VLAN_PORT_TYPE_S	S-port
VTSS_VLAN_PORT_TYPE_S_CUSTOM	S-port using alternative Ethernet Type

Definition at line 654 of file vtss_l2_api.h.

8.16.4.3 vtss_vlan_tx_tag_t

enum [vtss_vlan_tx_tag_t](#)

VLAN Tx tag type.

Enumerator

VTSS_VLAN_TX_TAG_PORT	Egress tagging determined by VLAN port configuration
VTSS_VLAN_TX_TAG_DISABLE	Egress tagging disabled
VTSS_VLAN_TX_TAG_ENABLE	Egress tagging enabled

Definition at line 778 of file vtss_l2_api.h.

8.16.4.4 vtss_vce_type_t

enum [vtss_vce_type_t](#)

VCE frame type.

Enumerator

VTSS_VCE_TYPE_ANY	Any frame type
VTSS_VCE_TYPE_ETYPE	Ethernet Type
VTSS_VCE_TYPE_LLC	LLC
VTSS_VCE_TYPE_SNAP	SNAP
VTSS_VCE_TYPE_IPV4	IPv4
VTSS_VCE_TYPE_IPV6	IPv6

Definition at line 909 of file vtss_l2_api.h.

8.16.4.5 vtss_mirror_tag_t

```
enum vtss_mirror_tag_t
```

Mirror port configuration.

Enumerator

VTSS_MIRROR_TAG_NONE	No mirror tag is added
VTSS_MIRROR_TAG_C	C-tag is added
VTSS_MIRROR_TAG_S	S-tag is added
VTSS_MIRROR_TAG_S_CUSTOM	Custom S-tag is added

Definition at line 1671 of file vtss_l2_api.h.

8.16.4.6 vtss_eps_port_type_t

```
enum vtss_eps_port_type_t
```

Port protection type.

Enumerator

VTSS_EPS_PORT_1_PLUS_1	1+1 protection
VTSS_EPS_PORT_1_FOR_1	1:1 protection

Definition at line 2134 of file vtss_l2_api.h.

8.16.4.7 vtss_eps_selector_t

```
enum vtss_eps_selector_t
```

EPS selector.

Enumerator

VTSS_EPS_SELECTOR_WORKING	Select working port
VTSS_EPS_SELECTOR_PROTECTION	Select protection port

Definition at line 2176 of file vtss_l2_api.h.

8.16.4.8 vtss_erps_state_t

enum [vtss_erps_state_t](#)

ERPS state.

Enumerator

VTSS_ERPS_STATE_FORWARDING	Forwarding
VTSS_ERPS_STATE_DISCARDING	Discarding

Definition at line 2266 of file vtss_l2_api.h.

8.16.4.9 vtss_vstax_topology_type_t

enum [vtss_vstax_topology_type_t](#)

VStaX topology type.

Enumerator

VTSS_VSTAX_TOPOLOGY_CHAIN	Chain topology
VTSS_VSTAX_TOPOLOGY_RING	Ring topology

Definition at line 2419 of file vtss_l2_api.h.

8.16.5 Function Documentation

8.16.5.1 vtss_mac_table_add()

```
vtss_rc vtss_mac_table_add (
    const vtss_inst_t inst,
    const vtss_mac_table_entry_t *const entry )
```

Add MAC address entry.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>entry</i>	[IN] MAC address entry structure.

Returns

Return code.

8.16.5.2 vtss_mac_table_del()

```
vtss_rc vtss_mac_table_del (
    const vtss_inst_t inst,
    const vtss_vid_mac_t *const vid_mac )
```

Delete MAC address entry.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>vid_mac</i>	[IN] VLAN ID and MAC address structure.

Returns

Return code.

8.16.5.3 vtss_mac_table_get()

```
vtss_rc vtss_mac_table_get (
    const vtss_inst_t inst,
    const vtss_vid_mac_t *const vid_mac,
    vtss_mac_table_entry_t *const entry )
```

Get MAC address entry.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>vid_mac</i>	[IN] VLAN ID and MAC address.
<i>entry</i>	[OUT] MAC address entry.

Returns

Return code.

8.16.5.4 vtss_mac_table_get_next()

```
vtss_rc vtss_mac_table_get_next (
    const vtss_inst_t inst,
    const vtss_vid_mac_t *const vid_mac,
    vtss_mac_table_entry_t *const entry )
```

Lookup next MAC address entry.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>vid_mac</i>	[IN] VLAN ID and MAC address.
<i>entry</i>	[OUT] MAC address entry.

Returns

Return code.

8.16.5.5 vtss_mac_table_age_time_get()

```
vtss_rc vtss_mac_table_age_time_get (
    const vtss_inst_t inst,
    vtss_mac_table_age_time_t *const age_time )
```

Get MAC address table age time.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>age_time</i>	[OUT] MAC age time in seconds. Value zero disables aging.

Returns

Return code.

8.16.5.6 vtss_mac_table_age_time_set()

```
vtss_rc vtss_mac_table_age_time_set (
    const vtss_inst_t inst,
    const vtss_mac_table_age_time_t age_time )
```

Set MAC address table age time.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>age_time</i>	[IN] MAC age time in seconds. Value zero disables aging.

Returns

Return code.

8.16.5.7 vtss_mac_table_age()

```
vtss_rc vtss_mac_table_age (
    const vtss_inst_t inst )
```

Do age scan of the MAC address table. This should be done periodically with interval T/2, where T is the age timer.

Parameters

<i>inst</i>	[IN] Target instance reference.
-------------	---------------------------------

Returns

Return code.

8.16.5.8 vtss_mac_table_vlan_age()

```
vtss_rc vtss_mac_table_vlan_age (
    const vtss_inst_t inst,
    const vtss_vid_t vid )
```

Do VLAN specific age scan of the MAC address table.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>vid</i>	[IN] VLAN ID.

Returns

Return code.

8.16.5.9 vtss_mac_table_flush()

```
vtss_rc vtss_mac_table_flush (
    const vtss_inst_t inst )
```

Flush MAC address table, i.e. remove all unlocked entries.

Parameters

<i>inst</i>	[IN] Target instance reference.
-------------	---------------------------------

Returns

Return code.

8.16.5.10 vtss_mac_table_port_flush()

```
vtss_rc vtss_mac_table_port_flush (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no )
```

Delete MAC address entries learned on port.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.

Returns

Return code.

8.16.5.11 vtss_mac_table_vlan_flush()

```
vtss_rc vtss_mac_table_vlan_flush (
    const vtss_inst_t inst,
    const vtss_vid_t vid )
```

Delete MAC address entries learned on VLAN ID.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>vid</i>	[IN] VLAN ID.

Returns

Return code.

8.16.5.12 vtss_mac_table_vlan_port_flush()

```
vtss_rc vtss_mac_table_vlan_port_flush (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_vid_t vid )
```

Delete MAC address entries learned on port and VLAN ID.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>vid</i>	[IN] VLAN ID.

Returns

Return code.

8.16.5.13 vtss_mac_table_upsid_flush()

```
vtss_rc vtss_mac_table_upsid_flush (
    const vtss_inst_t inst,
    const vtss_vstax_upsid_t upsid )
```

Delete MAC address entries learned on UPSID.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>upsid</i>	[IN] UPSID (Unit Port Set Identifier).

Returns

Return code.

8.16.5.14 vtss_mac_table_upsid_upspn_flush()

```
vtss_rc vtss_mac_table_upsid_upspn_flush (
    const vtss_inst_t inst,
```

```
const vtss_vstax_upsid_t upsid,
const vtss_vstax_upspn_t upspn )
```

Delete MAC address entries learned on (UPSID, UPSPN).

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>upsid</i>	[IN] UPSID (Unit Port Set Identifier).
<i>upspn</i>	[IN] UPSPN (Unit Port Set Port Number).

Returns

Return code.

8.16.5.15 vtss_mac_table_glag_add()

```
vtss_rc vtss_mac_table_glag_add (
    const vtss_inst_t inst,
    const vtss_mac_table_entry_t *const entry,
    const vtss_glag_no_t glag_no )
```

Learn MAC address entry on GLAG.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>entry</i>	[IN] MAC address entry structure (destination set is ignored)
<i>glag_no</i>	[IN] GLAG number.

Returns

Return code.

8.16.5.16 vtss_mac_table_glag_flush()

```
vtss_rc vtss_mac_table_glag_flush (
    const vtss_inst_t inst,
    const vtss_glag_no_t glag_no )
```

Delete MAC address entries learned on GLAG.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>glag_no</i>	[IN] GLAG number.

Returns

Return code.

8.16.5.17 vtss_mac_table_vlan_glag_flush()

```
vtss_rc vtss_mac_table_vlan_glag_flush (
    const vtss_inst_t inst,
    const vtss_glag_no_t glag_no,
    const vtss_vid_t vid )
```

Delete MAC address entries learned on GLAG and VID.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>glag_no</i>	[IN] GLAG number.
<i>vid</i>	[IN] VLAN ID.

Returns

Return code.

8.16.5.18 vtss_mac_table_status_get()

```
vtss_rc vtss_mac_table_status_get (
    const vtss_inst_t inst,
    vtss_mac_table_status_t *const status )
```

Get MAC address table status.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>status</i>	[OUT] MAC address table status.

Returns

Return code.

8.16.5.19 vtss_learn_port_mode_get()

```
vtss_rc vtss_learn_port_mode_get (
    const vtss_inst_t inst,
```

```
const vtss_port_no_t port_no,
vtss_learn_mode_t *const mode )
```

Get the learn mode for a port.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>mode</i>	[OUT] Learn mode.

Returns

Return code.

8.16.5.20 vtss_learn_port_mode_set()

```
vtss_rc vtss_learn_port_mode_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_learn_mode_t *const mode )
```

Set the learn mode for a port.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>mode</i>	[IN] Learn mode.

Returns

Return code.

8.16.5.21 vtss_port_state_get()

```
vtss_rc vtss_port_state_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    BOOL *const state )
```

Get port operational state.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>state</i>	[OUT] Port state, TRUE if link is up.

Generated by Doxygen

Returns

Return code.

8.16.5.22 vtss_port_state_set()

```
vtss_rc vtss_port_state_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    BOOL state )
```

Set port operational state.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>state</i>	[IN] Port state, TRUE if link is up.

Returns

Return code.

8.16.5.23 vtss_stp_port_state_get()

```
vtss_rc vtss_stp_port_state_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_stp_state_t *const state )
```

Get Spanning Tree state for a port.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>state</i>	[OUT] STP state.

Returns

Return code.

8.16.5.24 vtss_stp_port_state_set()

```
vtss_rc vtss_stp_port_state_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_stp_state_t state )
```

Set Spanning Tree state for a port.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>state</i>	[IN] STP state.

Returns

Return code.

8.16.5.25 vtss_mstp_vlan_msti_get()

```
vtss_rc vtss_mstp_vlan_msti_get (
    const vtss_inst_t inst,
    const vtss_vid_t vid,
    vtss_msti_t *const msti )
```

Get MSTP instance mapping for a VLAN.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>vid</i>	[IN] VLAN ID.
<i>msti</i>	[OUT] MSTP instance.

Returns

Return code.

8.16.5.26 vtss_mstp_vlan_msti_set()

```
vtss_rc vtss_mstp_vlan_msti_set (
    const vtss_inst_t inst,
    const vtss_vid_t vid,
    const vtss_msti_t msti )
```

Set MSTP instance mapping for a VLAN.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>vid</i>	[IN] VLAN ID.
<i>msti</i>	[IN] MSTP instance.

Returns

Return code.

8.16.5.27 vtss_mstp_port_msti_state_get()

```
vtss_rc vtss_mstp_port_msti_state_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_msti_t msti,
    vtss_stp_state_t *const state )
```

Get MSTP state for a port and MSTP instance.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>msti</i>	[IN] MSTP instance.
<i>state</i>	[OUT] MSTP state.

Returns

Return code.

8.16.5.28 vtss_mstp_port_msti_state_set()

```
vtss_rc vtss_mstp_port_msti_state_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_msti_t msti,
    const vtss_stp_state_t state )
```

Set MSTP state for a port and MSTP instance.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>msti</i>	[IN] MSTP instance.
<i>state</i>	[IN] MSTP state.

Returns

Return code.

8.16.5.29 vtss_vlan_conf_get()

```
vtss_rc vtss_vlan_conf_get (
    const vtss_inst_t inst,
    vtss_vlan_conf_t *const conf )
```

Get VLAN configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>conf</i>	[OUT] VLAN configuration structure.

Returns

Return code.

8.16.5.30 vtss_vlan_conf_set()

```
vtss_rc vtss_vlan_conf_set (
    const vtss_inst_t inst,
    const vtss_vlan_conf_t *const conf )
```

Set VLAN configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>conf</i>	[IN] VLAN configuration structure.

Returns

Return code.

8.16.5.31 vtss_vlan_port_conf_get()

```
vtss_rc vtss_vlan_port_conf_get (
    const vtss_inst_t inst,
```

```
const vtss_port_no_t port_no,
vtss_vlan_port_conf_t *const conf )
```

Get VLAN mode for port.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>conf</i>	[OUT] VLAN port configuration structure.

Returns

Return code.

8.16.5.32 vtss_vlan_port_conf_set()

```
vtss_rc vtss_vlan_port_conf_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_vlan_port_conf_t *const conf )
```

Set VLAN mode for port.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>conf</i>	[IN] VLAN port configuration structure.

Returns

Return code.

8.16.5.33 vtss_vlan_port_members_get()

```
vtss_rc vtss_vlan_port_members_get (
    const vtss_inst_t inst,
    const vtss_vid_t vid,
    BOOL member[VTSS_PORT_ARRAY_SIZE] )
```

Get VLAN membership.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>vid</i>	[IN] VLAN ID.
<i>member</i>	[OUT] VLAN port member list.

Returns

Return code.

8.16.5.34 vtss_vlan_port_members_set()

```
vtss_rc vtss_vlan_port_members_set (
    const vtss_inst_t inst,
    const vtss_vid_t vid,
    const BOOL member[VTSS_PORT_ARRAY_SIZE] )
```

Set VLAN membership.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>vid</i>	[IN] VLAN ID.
<i>member</i>	[IN] VLAN port member list.

Returns

Return code.

8.16.5.35 vtss_vlan_vid_conf_get()

```
vtss_rc vtss_vlan_vid_conf_get (
    const vtss_inst_t inst,
    const vtss_vid_t vid,
    vtss_vlan_vid_conf_t *const conf )
```

Get VLAN ID configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>vid</i>	[IN] VLAN ID.
<i>conf</i>	[OUT] VLAN configuration.

Returns

Return code.

8.16.5.36 vtss_vlan_vid_conf_set()

```
vtss_rc vtss_vlan_vid_conf_set (
    const vtss_inst_t inst,
    const vtss_vid_t vid,
    const vtss_vlan_vid_conf_t *const conf )
```

Set VLAN ID configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>vid</i>	[IN] VLAN ID.
<i>conf</i>	[IN] VLAN configuration.

Returns

Return code.

8.16.5.37 vtss_vlan_tx_tag_get()

```
vtss_rc vtss_vlan_tx_tag_get (
    const vtss_inst_t inst,
    const vtss_vid_t vid,
    vtss_vlan_tx_tag_t tx_tag[VTSS_PORT_ARRAY_SIZE] )
```

Get VLAN Tx tagging configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>vid</i>	[IN] VLAN ID.
<i>tx_tag</i>	[OUT] Tx tagging list.

Returns

Return code.

8.16.5.38 vtss_vlan_tx_tag_set()

```
vtss_rc vtss_vlan_tx_tag_set (
    const vtss_inst_t inst,
    const vtss_vid_t vid,
    const vtss_vlan_tx_tag_t tx_tag[VTSS_PORT_ARRAY_SIZE] )
```

Get VLAN Tx tagging configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>vid</i>	[IN] VLAN ID.
<i>tx_tag</i>	[IN] Tx tagging list.

Returns

Return code.

8.16.5.39 vtss_vcl_port_conf_get()

```
vtss_rc vtss_vcl_port_conf_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_vcl_port_conf_t *const conf )
```

Get VCL port configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>conf</i>	[OUT] VCL port configuration structure.

Returns

Return code.

8.16.5.40 vtss_vcl_port_conf_set()

```
vtss_rc vtss_vcl_port_conf_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_vcl_port_conf_t *const conf )
```

Set VCL port configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>conf</i>	[IN] VCL port configuration structure.

Returns

Return code.

8.16.5.41 vtss_vce_init()

```
vtss_rc vtss_vce_init (
    const vtss_inst_t inst,
    const vtss_vce_type_t type,
    vtss_vce_t *const vce )
```

Initialize VCE to default values.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>type</i>	[IN] VCE type.
<i>vce</i>	[OUT] VCE structure.

Returns

Return code.

8.16.5.42 vtss_vce_add()

```
vtss_rc vtss_vce_add (
    const vtss_inst_t inst,
    const vtss_vce_id_t vce_id,
    const vtss_vce_t *const vce )
```

Add/modify VCE.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>vce_id</i>	[IN] VCE ID. The VCE will be added before the entry with this ID. VTSS_VCE_ID_LAST is reserved for inserting last.
<i>vce</i>	[IN] VCE structure.

Returns

Return code.

8.16.5.43 vtss_vce_del()

```
vtss_rc vtss_vce_del (
    const vtss_inst_t inst,
    const vtss_vce_id_t vce_id )
```

Delete VCE.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>vce_id</i>	[IN] VCE ID.

Returns

Return code.

8.16.5.44 vtss_vlan_trans_group_add()

```
vtss_rc vtss_vlan_trans_group_add (
    const vtss_inst_t inst,
    const u16 group_id,
    const vtss_vid_t vid,
    const vtss_vid_t trans_vid )
```

Create VLAN Translation Group entry.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>group_id</i>	[IN] Group ID.
<i>vid</i>	[IN] VLAN ID.
<i>trans_vid</i>	[IN] Translated VLAN ID.

Returns

Return code.

8.16.5.45 vtss_vlan_trans_group_del()

```
vtss_rc vtss_vlan_trans_group_del (
    const vtss_inst_t inst,
    const u16 group_id,
    const vtss_vid_t vid )
```

Delete VLAN Translation Group entry.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>group_id</i>	[IN] Group ID.
<i>vid</i>	[IN] VLAN ID.

Returns

Return code.

8.16.5.46 vtss_vlan_trans_group_get()

```
vtss_rc vtss_vlan_trans_group_get (
    const vtss_inst_t inst,
    vtss_vlan_trans_grp2vlan_conf_t * conf,
    BOOL next )
```

Get VLAN Translation Group entry.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>conf</i>	[INOUT] Pointer to vtss_vlan_trans_grp2vlan_conf_t . Input group_id in the conf structure
<i>next</i>	[IN] Flag to indicate next entry.

Returns

Return code.

8.16.5.47 vtss_vlan_trans_group_to_port_set()

```
vtss_rc vtss_vlan_trans_group_to_port_set (
    const vtss_inst_t inst,
    const vtss_vlan_trans_port2grp_conf_t * conf )
```

Associate VLAN Translation Group entries to a port_list. Only one port can be part of one group not multiple groups.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>conf</i>	[IN] Pointer to vtss_vlan_trans_port2grp_conf_t .

Returns

Return code.

8.16.5.48 vtss_vlan_trans_group_to_port_get()

```
vtss_rc vtss_vlan_trans_group_to_port_get (
    const vtss_inst_t inst,
    vtss_vlan_trans_port2grp_conf_t * conf,
    BOOL next )
```

VLAN Translation function to fetch all ports for a group.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>conf</i>	[INOUT] Pointer to vtss_vlan_trans_port2grp_conf_t .
<i>next</i>	[IN] Flag to indicate next entry.

Returns

Return code.

8.16.5.49 vtss_isolated_vlan_get()

```
vtss_rc vtss_isolated_vlan_get (
    const vtss_inst_t inst,
    const vtss_vid_t vid,
    BOOL *const isolated )
```

Get enable/disable port isolation for VLAN.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>vid</i>	[IN] VLAN ID.
<i>isolated</i>	[OUT] VLAN isolation enable/disable option.

Returns

Return code.

8.16.5.50 vtss_isolated_vlan_set()

```
vtss_rc vtss_isolated_vlan_set (
    const vtss_inst_t inst,
    const vtss_vid_t vid,
    const BOOL isolated )
```

Set enable/disable port isolation for VLAN.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>vid</i>	[IN] VLAN ID.
<i>isolated</i>	[IN] VLAN isolation enable/disable option.

Returns

Return code.

8.16.5.51 vtss_isolated_port_members_get()

```
vtss_rc vtss_isolated_port_members_get (
    const vtss_inst_t inst,
    BOOL member[VTSS_PORT_ARRAY_SIZE] )
```

Get the isolated port member set.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>member</i>	[OUT] Isolated port member list.

Returns

Return code.

8.16.5.52 vtss_isolated_port_members_set()

```
vtss_rc vtss_isolated_port_members_set (
    const vtss_inst_t inst,
    const BOOL member[VTSS_PORT_ARRAY_SIZE] )
```

Set the isolated port member set.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>member</i>	[IN] Isolated port member list.

Returns

Return code.

8.16.5.53 vtss_pvlan_port_members_get()

```
vtss_rc vtss_pvlan_port_members_get (
    const vtss_inst_t inst,
    const vtss_pvlan_no_t pvlan_no,
    BOOL member[VTSS_PORT_ARRAY_SIZE] )
```

Get Private VLAN membership.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>pvlan_no</i>	[IN] Private VLAN group number.
<i>member</i>	[OUT] Private VLAN port member list.

Returns

Return code.

8.16.5.54 vtss_pvlan_port_members_set()

```
vtss_rc vtss_pvlan_port_members_set (
    const vtss_inst_t inst,
    const vtss_pvlan_no_t pvlan_no,
    const BOOL member[VTSS_PORT_ARRAY_SIZE] )
```

Set Private VLAN membership.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>pvlan_no</i>	[IN] Private VLAN group number.
<i>member</i>	[IN] Private VLAN port member list.

Returns

Return code.

8.16.5.55 vtss_apvlan_port_members_get()

```
vtss_rc vtss_apvlan_port_members_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    BOOL member[VTSS_PORT_ARRAY_SIZE] )
```

Get Asymmetric Private VLAN membership.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Ingress port.
<i>member</i>	[OUT] Asymmetric Private VLAN port member list.

Returns

Return code.

8.16.5.56 vtss_apvlan_port_members_set()

```
vtss_rc vtss_apvlan_port_members_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const BOOL member[VTSS_PORT_ARRAY_SIZE] )
```

Set Asymmetric Private VLAN membership.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Ingress port.
<i>member</i>	[IN] Asymmetric Private VLAN port member list.

Returns

Return code.

8.16.5.57 vtss_dgroup_port_conf_get()

```
vtss_rc vtss_dgroup_port_conf_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_dgroup_port_conf_t *const conf )
```

Get Destination Group configuration for port.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>conf</i>	[OUT] Destination group port configuration structure.

Returns

Return code.

8.16.5.58 vtss_dgroup_port_conf_set()

```
vtss_rc vtss_dgroup_port_conf_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_dgroup_port_conf_t *const conf )
```

Set Destination Group configuration for port.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>conf</i>	[IN] Destination group port configuration structure.

Returns

Return code.

8.16.5.59 vtss_sflow_port_conf_get()

```
vtss_rc vtss_sflow_port_conf_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_sflow_port_conf_t *const conf )
```

Get port sFlow configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number (a.k.a. data source).
<i>conf</i>	[OUT] sFlow sampler configuration.

Returns

Return code.

8.16.5.60 vtss_sf_low_port_conf_set()

```
vtss_rc vtss_sf_low_port_conf_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_sf_low_port_conf_t *const conf )
```

Set port sFlow configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number (a.k.a. data source).
<i>conf</i>	[IN] sFlow sampler configuration.

Returns

Return code.

8.16.5.61 vtss_sf_low_sampling_rate_convert()

```
vtss_rc vtss_sf_low_sampling_rate_convert (
    const vtss_inst_t inst,
    const BOOL power2,
    const u32 rate_in,
    u32 *const rate_out )
```

Convert desired sample rate to supported sample rate.

Since it may not be possible to realize all desired sample rates in H/W, this function can be used to query for an actual sample rate given a desired sample rate.

If the sFlow application code wishes to support more than one sampler instance per port, it will have to use only powers of two for the sampling rate. To obtain powers of two sampling rates, given an arbitrary input sampling rate, set power2 to TRUE, otherwise set it to FALSE.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>power2</i>	[IN] Only return sampling rates in powers of two.
<i>rate_in</i>	[IN] Desired sample rate
<i>rate_out</i>	[OUT] Realizable sample rate

Returns

Return code.

8.16.5.62 vtss_aggr_port_members_get()

```
vtss_rc vtss_aggr_port_members_get (
    const vtss_inst_t inst,
    const vtss_aggr_no_t aggr_no,
    BOOL member[VTSS_PORT_ARRAY_SIZE] )
```

Get aggregation port members.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>aggr_no</i>	[IN] Aggregation number.
<i>member</i>	[OUT] Aggregation port member list.

Returns

Return code.

8.16.5.63 vtss_aggr_port_members_set()

```
vtss_rc vtss_aggr_port_members_set (
    const vtss_inst_t inst,
    const vtss_aggr_no_t aggr_no,
    const BOOL member[VTSS_PORT_ARRAY_SIZE] )
```

Set aggregation port members.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>aggr_no</i>	[IN] Aggregation number.
<i>member</i>	[IN] Aggregation port member list.

Returns

Return code.

8.16.5.64 vtss_aggr_mode_get()

```
vtss_rc vtss_aggr_mode_get (
    const vtss_inst_t inst,
    vtss_aggr_mode_t *const mode )
```

Get aggregation traffic distribution mode.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>mode</i>	[OUT] Distribution mode structure.

Returns

Return code.

8.16.5.65 vtss_aggr_mode_set()

```
vtss_rc vtss_aggr_mode_set (
    const vtss_inst_t inst,
    const vtss_aggr_mode_t *const mode )
```

Set aggregation traffic distribution mode.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>mode</i>	[IN] Distribution mode structure.

Returns

Return code.

8.16.5.66 vtss_aggr_glag_members_get()

```
vtss_rc vtss_aggr_glag_members_get (
    const vtss_inst_t inst,
```

```
const vtss_glag_no_t glag_no,
BOOL member[VTSS_PORT_ARRAY_SIZE] )
```

Get global aggregation port members.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>glag_no</i>	[IN] GLAG number.
<i>member</i>	[OUT] GLAG port member list.

Returns

Return code.

8.16.5.67 vtss_vstax_glag_get()

```
vtss_rc vtss_vstax_glag_get (
    const vtss_inst_t inst,
    const vtss_glag_no_t glag_no,
    const vtss_vstax_glag_entry_t entry[VTSS_GLAG_PORT_ARRAY_SIZE] )
```

Get global aggregation port members.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>glag_no</i>	[IN] GLAG number.
<i>entry</i>	[OUT] GLAG entry list.

Returns

Return code.

8.16.5.68 vtss_vstax_glag_set()

```
vtss_rc vtss_vstax_glag_set (
    const vtss_inst_t inst,
    const vtss_glag_no_t glag_no,
    const vtss_vstax_glag_entry_t entry[VTSS_GLAG_PORT_ARRAY_SIZE] )
```

Get global aggregation port members.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>glag_no</i>	[IN] GLAG number.
<i>entry</i>	[IN] GLAG entry list.

Returns

Return code.

8.16.5.69 vtss_mirror_conf_get()

```
vtss_rc vtss_mirror_conf_get (
    const vtss_inst_t inst,
    vtss_mirror_conf_t *const conf )
```

Get the mirror configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>conf</i>	[OUT] Mirror configuration.

Returns

Return code.

8.16.5.70 vtss_mirror_conf_set()

```
vtss_rc vtss_mirror_conf_set (
    const vtss_inst_t inst,
    const vtss_mirror_conf_t *const conf )
```

Set the mirror configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>conf</i>	[IN] Mirror configuration.

Returns

Return code.

8.16.5.71 vtss_mirror_monitor_port_get()

```
vtss_rc vtss_mirror_monitor_port_get (
    const vtss_inst_t inst,
    vtss_port_no_t *const port_no )
```

Get the mirror monitor port.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[OUT] Port number.

Returns

Return code.

8.16.5.72 vtss_mirror_monitor_port_set()

```
vtss_rc vtss_mirror_monitor_port_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no )
```

Set the mirror monitor port.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number or VTSS_PORT_NO_NONE.

Returns

Return code.

8.16.5.73 vtss_mirror_ingress_ports_get()

```
vtss_rc vtss_mirror_ingress_ports_get (
    const vtss_inst_t inst,
    BOOL member[VTSS_PORT_ARRAY_SIZE] )
```

Get the mirror ingress ports.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>member</i>	[OUT] Port member list. If a port is enabled in this array, frames received on the port are mirrored.

Returns

Return code.

8.16.5.74 vtss_mirror_ingress_ports_set()

```
vtss_rc vtss_mirror_ingress_ports_set (
    const vtss_inst_t inst,
    const BOOL member[VTSS_PORT_ARRAY_SIZE] )
```

Set the mirror ingress ports.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>member</i>	[IN] Port member list. If a port is enabled in this array, frames received on the port are mirrored.

Returns

Return code.

8.16.5.75 vtss_mirror_egress_ports_get()

```
vtss_rc vtss_mirror_egress_ports_get (
    const vtss_inst_t inst,
    BOOL member[VTSS_PORT_ARRAY_SIZE] )
```

Get the mirror egress ports.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>member</i>	[OUT] Port member list. If a port is enabled in this array, frames transmitted on the port are mirrored.

Returns

Return code.

8.16.5.76 vtss_mirror_egress_ports_set()

```
vtss_rc vtss_mirror_egress_ports_set (
    const vtss_inst_t inst,
    const BOOL member[VTSS_PORT_ARRAY_SIZE] )
```

Set the mirror egress ports.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>member</i>	[IN] Port member list. If a port is enabled in this array, frames transmitted on the port are mirrored.

Returns

Return code.

8.16.5.77 vtss_mirror_cpu_ingress_get()

```
vtss_rc vtss_mirror_cpu_ingress_get (
    const vtss_inst_t inst,
    BOOL * member )
```

Get the mirror CPU ingress.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>member</i>	[OUT] CPU member. If a CPU is enabled, frames received to the CPU port are mirrored.

Returns

Return code.

8.16.5.78 vtss_mirror_cpu_ingress_set()

```
vtss_rc vtss_mirror_cpu_ingress_set (
    const vtss_inst_t inst,
    const BOOL member )
```

Set CPU ingress mirroring.

Enabling CPU ingress mirroring means that frames destined for the CPU are mirrored to the mirror port.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>member</i>	[IN] CPU member. If a CPU is enabled, frames received by the CPU port are mirrored.

Returns

Return code.

8.16.5.79 vtss_mirror_cpu_egress_get()

```
vtss_rc vtss_mirror_cpu_egress_get (
    const vtss_inst_t inst,
    BOOL * member )
```

Get the mirror CPU egress.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>member</i>	[OUT] CPU member. If a CPU is enabled, frames transmitted by the CPU port are mirrored.

Returns

Return code.

8.16.5.80 vtss_mirror_cpu_egress_set()

```
vtss_rc vtss_mirror_cpu_egress_set (
    const vtss_inst_t inst,
    const BOOL member )
```

Set the mirror CPU egress.

Enabling CPU egress mirroring means that frames transmitted by the CPU are mirrored to the mirror port.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>member</i>	[IN] CPU member. If a CPU is enabled, frames transmitted by the CPU the port are mirrored.

Returns

Return code.

8.16.5.81 vtss_uc_flood_members_get()

```
vtss_rc vtss_uc_flood_members_get (
    const vtss_inst_t inst,
    BOOL member[VTSS_PORT_ARRAY_SIZE] )
```

Get unicast flood members.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>member</i>	[OUT] Port member list.

Returns

Return code.

8.16.5.82 vtss_uc_flood_members_set()

```
vtss_rc vtss_uc_flood_members_set (
    const vtss_inst_t inst,
    const BOOL member[VTSS_PORT_ARRAY_SIZE] )
```

Set unicast flood members.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>member</i>	[IN] Port member list.

Returns

Return code.

8.16.5.83 vtss_mc_flood_members_get()

```
vtss_rc vtss_mc_flood_members_get (
    const vtss_inst_t inst,
    BOOL member[VTSS_PORT_ARRAY_SIZE] )
```

Get multicast flood members.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>member</i>	[OUT] Port member list.

Returns

Return code.

8.16.5.84 vtss_mc_flood_members_set()

```
vtss_rc vtss_mc_flood_members_set (
    const vtss_inst_t inst,
    const BOOL member[VTSS_PORT_ARRAY_SIZE] )
```

Set multicast flood members.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>member</i>	[IN] Port member list.

Returns

Return code.

8.16.5.85 vtss_ipv4_mc_flood_members_get()

```
vtss_rc vtss_ipv4_mc_flood_members_get (
    const vtss_inst_t inst,
    BOOL member[VTSS_PORT_ARRAY_SIZE] )
```

Get IPv4 multicast flood members.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>member</i>	[OUT] Port member list. Ports connected to IPv4 multicast routers should be enabled.

Returns

Return code.

8.16.5.86 vtss_ipv4_mc_flood_members_set()

```
vtss_rc vtss_ipv4_mc_flood_members_set (
    const vtss_inst_t inst,
    const BOOL member[VTSS_PORT_ARRAY_SIZE] )
```

Set IPv4 multicast flood members.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>member</i>	[IN] Port member list. Ports connected to IPv4 multicast routers should be enabled.

Returns

Return code.

8.16.5.87 vtss_ipv6_mc_flood_members_get()

```
vtss_rc vtss_ipv6_mc_flood_members_get (
    const vtss_inst_t inst,
    BOOL member[VTSS_PORT_ARRAY_SIZE] )
```

Get IPv6 multicast flood members.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>member</i>	[OUT] Port member list. Ports connected to IPv6 multicast routers should be enabled.

Returns

Return code.

8.16.5.88 vtss_ipv6_mc_flood_members_set()

```
vtss_rc vtss_ipv6_mc_flood_members_set (
    const vtss_inst_t inst,
    const BOOL member[VTSS_PORT_ARRAY_SIZE] )
```

Set IPv6 multicast flood members.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>member</i>	[IN] Port member list. Ports connected to IPv6 multicast routers should be enabled.

Returns

Return code.

8.16.5.89 vtss_ipv6_mc_ctrl_flood_get()

```
vtss_rc vtss_ipv6_mc_ctrl_flood_get (
    const vtss_inst_t inst,
    BOOL *const scope )
```

Get IPv6 multicast control flooding mode.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>scope</i>	[OUT] IPv6 multicast control flood. 0: Flood to all ports. 1: Flood to IPv6 multicast flood members.

Returns

Return code.

8.16.5.90 vtss_ipv6_mc_ctrl_flood_set()

```
vtss_rc vtss_ipv6_mc_ctrl_flood_set (
    const vtss_inst_t inst,
    const BOOL scope )
```

Set IPv6 multicast control flooding mode. This controls whether unknown Link-Local scope IPv6 multicasts (FF02::/16) are flooded to all ports or to the ports in the IPv6 multicast flood mask.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>scope</i>	[IN] IPv6 multicast control flood. 0: Flood to all ports. 1: Flood to IPv6 multicast flood members.

Returns

Return code.

8.16.5.91 vtss_eps_port_conf_get()

```
vtss_rc vtss_eps_port_conf_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_eps_port_conf_t *const conf )
```

Get EPS port configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Working port.
<i>conf</i>	[OUT] Protection configuration.

Returns

Return code.

8.16.5.92 vtss_eps_port_conf_set()

```
vtss_rc vtss_eps_port_conf_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_eps_port_conf_t *const conf )
```

Set EPS port configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Working port.
<i>conf</i>	[IN] Protection configuration.

Returns

Return code.

8.16.5.93 vtss_eps_port_selector_get()

```
vtss_rc vtss_eps_port_selector_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_eps_selector_t *const selector )
```

Get EPS port selector.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Working port.
<i>selector</i>	[OUT] Selector.

Returns

Return code.

8.16.5.94 vtss_eps_port_selector_set()

```
vtss_rc vtss_eps_port_selector_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_eps_selector_t selector )
```

Set EPS port selector.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Working port.
<i>selector</i>	[IN] Selector.

Returns

Return code.

8.16.5.95 vtss_erps_vlan_member_get()

```
vtss_rc vtss_erps_vlan_member_get (
    const vtss_inst_t inst,
    const vtss_erpit_t erpi,
    const vtss_vid_t vid,
    BOOL *const member )
```

Get ERPS member state for a VLAN.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>erpi</i>	[IN] ERPS instance.
<i>vid</i>	[IN] VLAN ID.
<i>member</i>	[OUT] Membership, TRUE if VLAN is included in ERPS instance.

Returns

Return code.

8.16.5.96 vtss_erps_vlan_member_set()

```
vtss_rc vtss_erps_vlan_member_set (
    const vtss_inst_t inst,
    const vtss_erpit_t erpi,
    const vtss_vid_t vid,
    const BOOL member )
```

Set ERPS member state for a VLAN.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>erpi</i>	[IN] ERPS instance.
<i>vid</i>	[IN] VLAN ID.
<i>member</i>	[IN] Membership, TRUE if VLAN is included in ERPS instance.

Returns

Return code.

8.16.5.97 vtss_erps_port_state_get()

```
vtss_rc vtss_erps_port_state_get (
    const vtss_inst_t inst,
    const vtss_erpi_t erpi,
    const vtss_port_no_t port_no,
    vtss_erps_state_t *const state )
```

Get ERPS state for ERPS instance and port.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>erpi</i>	[IN] ERPS instance.
<i>port_no</i>	[IN] Port number.
<i>state</i>	[OUT] ERPS state.

Returns

Return code.

8.16.5.98 vtss_erps_port_state_set()

```
vtss_rc vtss_erps_port_state_set (
    const vtss_inst_t inst,
    const vtss_erpi_t erpi,
    const vtss_port_no_t port_no,
    const vtss_erps_state_t state )
```

Set ERPS state for ERPS instance and port.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>erpi</i>	[IN] ERPS instance.
<i>state</i>	[IN] ERPS state.

Returns

Return code.

8.16.5.99 `vtss_vstax_conf_get()`

```
vtss_rc vtss_vstax_conf_get (
    const vtss_inst_t inst,
    vtss_vstax_conf_t *const conf )
```

Get VStaX configuration for switch.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>conf</i>	[OUT] VStaX configuration.

Returns

Return code.

8.16.5.100 `vtss_vstax_conf_set()`

```
vtss_rc vtss_vstax_conf_set (
    const vtss_inst_t inst,
    const vtss_vstax_conf_t *const conf )
```

Set VStaX configuration for switch.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>conf</i>	[IN] VStaX configuration.

Returns

Return code.

8.16.5.101 `vtss_vstax_port_conf_get()`

```
vtss_rc vtss_vstax_port_conf_get (
    const vtss_inst_t inst,
    const vtss_chip_no_t chip_no,
    const BOOL stack_port_a,
    vtss_vstax_port_conf_t *const conf )
```

Get VStaX configuration for port.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>chip_no</i>	[IN] Chip number (if multi-chip instance).
<i>stack_port_a</i>	[IN] Stack port A/B indication.
<i>conf</i>	[OUT] VStaX port configuration.

Returns

Return code.

8.16.5.102 vtss_vstax_port_conf_set()

```
vtss_rc vtss_vstax_port_conf_set (
    const vtss_inst_t inst,
    const vtss_chip_no_t chip_no,
    const BOOL stack_port_a,
    const vtss_vstax_port_conf_t *const conf )
```

Set VStaX configuration for port.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>chip_no</i>	[IN] Chip number (if multi-chip instance).
<i>stack_port_a</i>	[IN] Stack port A/B indication.
<i>conf</i>	[IN] VStaX port configuration.

Returns

Return code.

8.16.5.103 vtss_vstax_master_upsid_get()

```
vtss_rc vtss_vstax_master_upsid_get (
    const vtss_inst_t inst,
    vtss_vstax_upsid_t *const master_upsid )
```

Get UPSID of current master in stack.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>master_upsid</i>	[OUT] UPSID of current master or VTSS_VSTAX_UPSID_UNDEF if unknown.

Returns

Return code.

8.16.5.104 vtss_vstax_master_upsid_set()

```
vtss_rc vtss_vstax_master_upsid_set (
    const vtss_inst_t inst,
    const vtss_vstax_upsid_t master_upsid )
```

Set UPSID of current master in stack.

Whether this info is used or not by the API is chip-specific.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>master_upsid</i>	[IN] UPSID of current master or VTSS_VSTAX_UPSID_UNDEF if unknown.

Returns

Return code.

8.16.5.105 vtss_vstax_topology_set()

```
vtss_rc vtss_vstax_topology_set (
    const vtss_inst_t inst,
    const vtss_chip_no_t chip_no,
    const vtss_vstax_route_table_t * table )
```

Set stack topology.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>chip_no</i>	[IN] Chip number (if multi-chip instance).
<i>table</i>	[IN] Stack routing table.

Returns

Return code.

8.17 vtss_api/include/vtss_l3_api.h File Reference

L3 routing API.

```
#include <vtss/api/types.h>
```

Data Structures

- struct [vtss_l3_common_conf_t](#)
Common configurations for all routing legs.
- struct [vtss_l3_rleg_conf_t](#)
Router leg control structure.
- struct [vtss_l3_neighbour_t](#)
Neighbour entry.

Macros

- #define [VTSS_JR1_LPM_CNT](#) (1024u)
- #define [VTSS_LPM_CNT](#) [VTSS_JR1_LPM_CNT](#)
- #define [VTSS_JR1_ARP_CNT](#) (1024u)
- #define [VTSS_ARP_CNT](#) [VTSS_JR1_ARP_CNT](#)
- #define [VTSS_JR1_RLEG_CNT](#) (128u)
- #define [VTSS_RLEG_CNT](#) [VTSS_JR1_RLEG_CNT](#)
- #define [VTSS_ARP_IPV4_RELATIONS](#) [VTSS_ARP_CNT](#)
- #define [VTSS_ARP_IPV6_RELATIONS](#) [VTSS_ARP_CNT](#)

Typedefs

- typedef [u32 vtss_l3_rleg_id_t](#)
Router leg ID.
- typedef [u8 vtss_l3_vrid_t](#)
Virtual router identifier.

Enumerations

- enum [vtss_l3_rleg_common_mode_t](#){ [VTSS_ROUTING_RLEG_MAC_MODE_INVALID](#) = 0, [VTSS_ROUTING_RLEG_MAC_MODE_ARP](#) = 1 }
MAC addressing mode for routing legs.
- enum [vtss_l3_neighbour_type_t](#){ [VTSS_L3_NEIGHBOUR_TYPE_INVALID](#) = 0, [VTSS_L3_NEIGHBOUR_TYPE_ARP](#) = 1, [VTSS_L3_NEIGHBOUR_TYPE_NDP](#) = 2 }
Neighbour type.

Functions

- `vtss_rc vtss_l3_flush (const vtss_inst_t inst)`
Flush all L3 configurations.
- `vtss_rc vtss_l3_common_get (const vtss_inst_t inst, vtss_l3_common_conf_t *const conf)`
Get common router configuration.
- `vtss_rc vtss_l3_common_set (const vtss_inst_t inst, const vtss_l3_common_conf_t *const conf)`
Set common router configuration.
- `vtss_rc vtss_l3_rleg_get (const vtss_inst_t inst, u32 *cnt, vtss_l3_rleg_conf_t buf[VTSS_RLEG_CNT])`
Get all configured router leg.
- `vtss_rc vtss_l3_rleg_get_specific (const vtss_inst_t inst, vtss_vid_t vid, vtss_l3_rleg_conf_t *conf)`
Get a specific configured router leg.
- `vtss_rc vtss_l3_rleg_add (const vtss_inst_t inst, const vtss_l3_rleg_conf_t *const conf)`
Add a router leg on the given VLAN.
- `vtss_rc vtss_l3_rleg_update (const vtss_inst_t inst, const vtss_l3_rleg_conf_t *const conf)`
Update an existing router leg.
- `vtss_rc vtss_l3_rleg_del (const vtss_inst_t inst, const vtss_vid_t vlan)`
Delete a router leg associated with VLAN.
- `vtss_rc vtss_l3_route_get (const vtss_inst_t inst, u32 *cnt, vtss_routing_entry_t buf[VTSS_LPM_CNT])`
Get all configured routes.
- `vtss_rc vtss_l3_route_add (const vtss_inst_t inst, const vtss_routing_entry_t *const entry)`
Add an entry to the routing table.
- `vtss_rc vtss_l3_route_del (const vtss_inst_t inst, const vtss_routing_entry_t *const entry)`
Delete an entry from the routing table.
- `vtss_rc vtss_l3_neighbour_get (const vtss_inst_t inst, u32 *cnt, vtss_l3_neighbour_t buf[VTSS_ARP_CNT])`
Get all configured neighbours.
- `vtss_rc vtss_l3_neighbour_add (const vtss_inst_t inst, const vtss_l3_neighbour_t *const entry)`
Add a new entry to the neighbour cache.
- `vtss_rc vtss_l3_neighbour_del (const vtss_inst_t inst, const vtss_l3_neighbour_t *const entry)`
Delete an entry from the neighbour cache.
- `vtss_rc vtss_l3_counters_reset (const vtss_inst_t inst)`
Reset all routing leg statistics counters.
- `vtss_rc vtss_l3_counters_system_get (const vtss_inst_t inst, vtss_l3_counters_t *const counters)`
Get routing system counters.
- `vtss_rc vtss_l3_counters_rleg_get (const vtss_inst_t inst, const vtss_vid_t vlan, vtss_l3_counters_t *const counters)`
Get routing legs counters.
- `vtss_rc vtss_l3_counters_rleg_clear (const vtss_inst_t inst, const vtss_vid_t vlan)`
Clear routing legs counters.

8.17.1 Detailed Description

L3 routing API.

This header file describes L3 IPv4/IPv6 hardware assisted routing functions.

8.17.2 Macro Definition Documentation

8.17.2.1 VTSS_JR1_LPM_CNT

```
#define VTSS_JR1_LPM_CNT (1024u)
```

JR1 length of LPM table

Definition at line 113 of file vtss_l3_api.h.

8.17.2.2 VTSS_LPM_CNT

```
#define VTSS_LPM_CNT VTSS_JR1_LPM_CNT
```

Length of LPM table

Definition at line 114 of file vtss_l3_api.h.

8.17.2.3 VTSS_JR1_ARP_CNT

```
#define VTSS_JR1_ARP_CNT (1024u)
```

JR1 length of ARP table

Definition at line 116 of file vtss_l3_api.h.

8.17.2.4 VTSS_ARP_CNT

```
#define VTSS_ARP_CNT VTSS_JR1_ARP_CNT
```

Length of ARP table

Definition at line 117 of file vtss_l3_api.h.

8.17.2.5 VTSS_JR1_RLEG_CNT

```
#define VTSS_JR1_RLEG_CNT (128u)
```

JR1 length of RLEG table

Definition at line 119 of file vtss_l3_api.h.

8.17.2.6 VTSS_RLEG_CNT

```
#define VTSS_RLEG_CNT VTSS_JR1_RLEG_CNT
```

Length of RLEG table

Definition at line 120 of file vtss_l3_api.h.

8.17.2.7 VTSS_ARP_IPV4_RELATIONS

```
#define VTSS_ARP_IPV4_RELATIONS VTSS_ARP_CNT
```

Length of IPv4 ARP table

Definition at line 137 of file vtss_l3_api.h.

8.17.2.8 VTSS_ARP_IPV6_RELATIONS

```
#define VTSS_ARP_IPV6_RELATIONS VTSS_ARP_CNT
```

Length of IPv6 NDP table

Definition at line 140 of file vtss_l3_api.h.

8.17.3 Enumeration Type Documentation

8.17.3.1 vtss_l3_rleg_common_mode_t

```
enum vtss_l3_rleg_common_mode_t
```

MAC addressing mode for routing legs.

Enumerator

VTSS_ROUTING_RLEG_MAC_MODE_INVALID	The addressing mode has still not been configured
VTSS_ROUTING_RLEG_MAC_MODE_SINGLE	One common MAC address is used for all legs

Definition at line 153 of file vtss_l3_api.h.

8.17.3.2 vtss_l3_neighbour_type_t

enum `vtss_l3_neighbour_type_t`

Neighbour type.

Enumerator

<code>VTSS_L3_NEIGHBOUR_TYPE_INVALID</code>	Invalid entry.
<code>VTSS_L3_NEIGHBOUR_TYPE_ARP</code>	IPv4 Neighbour entry (ARP).
<code>VTSS_L3_NEIGHBOUR_TYPE_NDP</code>	IPv6 Neighbour entry (NDP).

Definition at line 217 of file vtss_l3_api.h.

8.17.4 Function Documentation

8.17.4.1 vtss_l3_flush()

```
vtss_rc vtss_l3_flush (
    const vtss_inst_t inst )
```

Flush all L3 configurations.

Parameters

<code>inst</code>	[IN] Target instance reference.
-------------------	---------------------------------

Returns

Return code.

8.17.4.2 vtss_l3_common_get()

```
vtss_rc vtss_l3_common_get (
    const vtss_inst_t inst,
    vtss_l3_common_conf_t *const conf )
```

Get common router configuration.

Parameters

<code>inst</code>	[IN] Target instance reference.
<code>conf</code>	[OUT] Common routing configurations.

Returns

Return code.

8.17.4.3 vtss_l3_common_set()

```
vtss_rc vtss_l3_common_set (
    const vtss_inst_t inst,
    const vtss_l3_common_conf_t *const conf )
```

Set common router configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>conf</i>	[OUT] Common routing configurations.

Returns

Return code.

8.17.4.4 vtss_l3_rleg_get()

```
vtss_rc vtss_l3_rleg_get (
    const vtss_inst_t inst,
    u32 * cnt,
    vtss_l3_rleg_conf_t buf[VTSS_RLEG_CNT] )
```

Get all configured router leg.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>cnt</i>	[OUT] Amount of entries copied to output buffer
<i>buf</i>	[OUT] Output buffer

Returns

Return code.

8.17.4.5 vtss_l3_rleg_get_specific()

```
vtss_rc vtss_l3_rleg_get_specific (
    const vtss_inst_t inst,
```

```
vtss_vid_t vid,
vtss_l3_rleg_conf_t * conf )
```

Get a specific configured router leg.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>vid</i>	[IN] VLAN ID of the router leg to get
<i>conf</i>	[OUT] Output buffer where the configuration is written

Returns

Return code.

8.17.4.6 vtss_l3_rleg_add()

```
vtss_rc vtss_l3_rleg_add (
    const vtss_inst_t inst,
    const vtss_l3_rleg_conf_t *const conf )
```

Add a router leg on the given VLAN.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>conf</i>	[IN] Routing leg configuration.

Returns

Return code.

8.17.4.7 vtss_l3_rleg_update()

```
vtss_rc vtss_l3_rleg_update (
    const vtss_inst_t inst,
    const vtss_l3_rleg_conf_t *const conf )
```

Update an existing router leg.

Will fail if an existing router leg with the same VLAN does not exists.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>conf</i>	[IN] Routing leg configuration.

Returns

Return code.

8.17.4.8 vtss_l3_rleg_del()

```
vtss_rc vtss_l3_rleg_del (
    const vtss_inst_t inst,
    const vtss_vid_t vlan )
```

Delete a router leg associated with VLAN.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>vlan</i>	[IN] VLAN to delete router leg from

Returns

Return code.

8.17.4.9 vtss_l3_route_get()

```
vtss_rc vtss_l3_route_get (
    const vtss_inst_t inst,
    u32 * cnt,
    vtss_routing_entry_t buf[VTSS_LPM_CNT] )
```

Get all configured routes.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>cnt</i>	[OUT] Amount of entries copied to output buffer
<i>buf</i>	[OUT] Output buffer

Returns

Return code.

8.17.4.10 vtss_l3_route_add()

```
vtss_rc vtss_l3_route_add (
    const vtss_inst_t inst,
    const vtss_routing_entry_t *const entry )
```

Add an entry to the routing table.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>entry</i>	[IN] Route to add

Returns

Return code.

8.17.4.11 vtss_l3_route_del()

```
vtss_rc vtss_l3_route_del (
    const vtss_inst_t inst,
    const vtss_routing_entry_t *const entry )
```

Delete an entry from the routing table.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>entry</i>	[IN] Entry to delete.

Returns

Return code.

8.17.4.12 vtss_l3_neighbour_get()

```
vtss_rc vtss_l3_neighbour_get (
    const vtss_inst_t inst,
    u32 * cnt,
    vtss_l3_neighbour_t buf[VTSS_ARP_CNT] )
```

Get all configured neighbours.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>cnt</i>	[OUT] Amount of entries copied to output buffer
<i>buf</i>	[OUT] Output buffer

Returns

Return code.

8.17.4.13 vtss_l3_neighbour_add()

```
vtss_rc vtss_l3_neighbour_add (
    const vtss_inst_t inst,
    const vtss_l3_neighbour_t *const entry )
```

Add a new entry to the neighbour cache.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>entry</i>	[IN] Entry to add.

Returns

Return code.

8.17.4.14 vtss_l3_neighbour_del()

```
vtss_rc vtss_l3_neighbour_del (
    const vtss_inst_t inst,
    const vtss_l3_neighbour_t *const entry )
```

Delete an entry from the neighbour cache.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>entry</i>	[IN] Entry to delete.

Returns

Return code.

8.17.4.15 vtss_l3_counters_reset()

```
vtss_rc vtss_l3_counters_reset (
    const vtss_inst_t inst )
```

Reset all routing leg statistics counters.

Parameters

<i>inst</i>	[IN] Target instance reference.
-------------	---------------------------------

Returns

Return code.

8.17.4.16 vtss_l3_counters_system_get()

```
vtss_rc vtss_l3_counters_system_get (
    const vtss_inst_t inst,
    vtss_l3_counters_t *const counters )
```

Get routing system counters.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>counters</i>	[OUT] Counters

Returns

Return code.

8.17.4.17 vtss_l3_counters_rleg_get()

```
vtss_rc vtss_l3_counters_rleg_get (
    const vtss_inst_t inst,
    const vtss_vid_t vlan,
    vtss_l3_counters_t *const counters )
```

Get routing legs counters.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>vlan</i>	[IN] Routing leg
<i>counters</i>	[OUT] Counters

Returns

Return code.

8.17.4.18 vtss_l3_counters_rleg_clear()

```
vtss_rc vtss_l3_counters_rleg_clear (
    const vtss_inst_t inst,
    const vtss_vid_t vlan )
```

Clear routing legs counters.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>vlan</i>	[IN] Routing leg

Returns

Return code.

8.18 vtss_api/include/vtss_mac10g_api.h File Reference

MAC10G API.

```
#include <vtss/api/types.h>
```

8.18.1 Detailed Description

MAC10G API.

8.19 vtss_api/include/vtss_misc_api.h File Reference

Miscellaneous API.

```
#include <vtss/api/types.h>
#include "vtss_init_api.h"
```

Data Structures

- struct `vtss_trace_conf_t`
Trace group configuration.
- struct `vtss_debug_info_t`
Debug information structure.
- struct `vtss_api_lock_t`
API lock structure.
- struct `vtss_debug_lock_t`
API debug lock structure.

- struct `vtss_chip_id_t`
Chip ID.
- struct `vtss_sgpio_port_conf_t`
SGPIO port configuration.
- struct `vtss_sgpio_conf_t`
SGPIO configuration for a group.
- struct `vtss_sgpio_port_data_t`
SGPIO read data for a port.
- struct `vtss_irq_conf_t`
Interrupt configuration options.
- struct `vtss_irq_status_t`
Interrupt status structure.
- struct `vtss_os_timestamp_t`
- struct `vtss_fan_conf_t`
Fan specifications.
- struct `vtss_eee_port_conf_t`
EEE port configuration.
- struct `vtss_eee_port_state_t`
EEE port state (JR only)
- struct `vtss_eee_port_counter_t`
EEE port counters (JR only)

Macros

- #define `VTSS_CHIP_NO_ALL` 0xffffffff
Special chip number value for showing information from all chips.
- #define `VTSS_GPIOS` 24
Number of GPIOs.
- #define `VTSS_GPIO_NO_START` 0
GPIO start number.
- #define `VTSS_GPIO_NO_END` (`VTSS_GPIO_NO_START`+`VTSS_GPIOS`)
GPIO end number.
- #define `VTSS_SGPIO_GROUPS` 2
Number of serial GPIO groups.
- #define `VTSS_SGPIO_PORTS` 32
Number of serial GPIO ports.
- #define `VTSS_OS_TIMESTAMP_TYPE` `vtss_os_timestamp_t`
- #define `VTSS_OS_TIMESTAMP`(`timestamp`)
- #define `VTSS_FAN_SPEED_MAX` 0x255
Maximum fan speed level (Fan runs at full speed)
- #define `VTSS_FAN_SPEED_MIN` 0x0
Minimum fan speed level (Fan is OFF)

Typedefs

- typedef void(* `vtss_debug_printf_t`) (const char *fmt,...)
Debug printf function.
- typedef `u32 vtss_gpio_no_t`
GPIO number.
- typedef `u32 vtss_sgpio_group_t`
Serial GPIO group.
- typedef `u32(* tod_get_ns_cnt_cb_t)` (`void`)
If the actual HW does not support time stamping, an external callback function can be set up to do the work.

Enumerations

- enum `vtss_trace_layer_t`{ `VTSS_TRACE_LAYER_AIL`, `VTSS_TRACE_LAYER_CIL`, `VTSS_TRACE_LAYER_COUNT` }

Trace group layer.

- enum `vtss_trace_group_t`{
`VTSS_TRACE_GROUP_DEFAULT`, `VTSS_TRACE_GROUP_PORT`, `VTSS_TRACE_GROUP_PHY`,
`VTSS_TRACE_GROUP_PACKET`,
`VTSS_TRACE_GROUP_AFI`, `VTSS_TRACE_GROUP_QOS`, `VTSS_TRACE_GROUP_L2`, `VTSS_TRACE_GROUP_L3`,
`VTSS_TRACE_GROUP_SECURITY`, `VTSS_TRACE_GROUP_EVC`, `VTSS_TRACE_GROUP_FDMA_NORMAL`,
`VTSS_TRACE_GROUP_FDMA_IRQ`,
`VTSS_TRACE_GROUP_REG_CHECK`, `VTSS_TRACE_GROUP MPLS`, `VTSS_TRACE_GROUP_HQOS`,
`VTSS_TRACE_GROUP_MACSEC`,
`VTSS_TRACE_GROUP_VCAP`, `VTSS_TRACE_GROUP_OAM`, `VTSS_TRACE_GROUP_TS`, `VTSS_TRACE_GROUP_COUM` }

Trace groups.

- enum `vtss_trace_level_t`{
`VTSS_TRACE_LEVEL_NONE`, `VTSS_TRACE_LEVEL_ERROR`, `VTSS_TRACE_LEVEL_INFO`, `VTSS_TRACE_LEVEL_DEBUG`,
`VTSS_TRACE_LEVEL_NOISE`, `VTSS_TRACE_LEVEL_COUNT` }

Trace levels.

- enum `vtss_debug_layer_t`{ `VTSS_DEBUG_LAYER_ALL`, `VTSS_DEBUG_LAYER_AIL`, `VTSS_DEBUG_LAYER_CIL` }

Debug layer.

- enum `vtss_debug_group_t`{
`VTSS_DEBUG_GROUP_ALL`, `VTSS_DEBUG_GROUP_INIT`, `VTSS_DEBUG_GROUP_MISC`, `VTSS_DEBUG_GROUP_PORT`,
`VTSS_DEBUG_GROUP_PORT_CNT`, `VTSS_DEBUG_GROUP_PHY`, `VTSS_DEBUG_GROUP_VLAN`,
`VTSS_DEBUG_GROUP_PVLAN`,
`VTSS_DEBUG_GROUP_MAC_TABLE`, `VTSS_DEBUG_GROUP_ACL`, `VTSS_DEBUG_GROUP_QOS`,
`VTSS_DEBUG_GROUP_AGGR`,
`VTSS_DEBUG_GROUP_GLAG`, `VTSS_DEBUG_GROUP_STP`, `VTSS_DEBUG_GROUP_MIRROR`,
`VTSS_DEBUG_GROUP_EVC`,
`VTSS_DEBUG_GROUP_ERPS`, `VTSS_DEBUG_GROUP_EPS`, `VTSS_DEBUG_GROUP_PACKET`,
`VTSS_DEBUG_GROUP_FDMA`,
`VTSS_DEBUG_GROUP_TS`, `VTSS_DEBUG_GROUP_PHY_TS`, `VTSS_DEBUG_GROUP_WM`, `VTSS_DEBUG_GROUP_LRM`,
`VTSS_DEBUG_GROUP_IPMC`, `VTSS_DEBUG_GROUP_STACK`, `VTSS_DEBUG_GROUP_CMEF`,
`VTSS_DEBUG_GROUP_HOST`,
`VTSS_DEBUG_GROUP_MPLS`, `VTSS_DEBUG_GROUP_MPLS_OAM`, `VTSS_DEBUG_GROUP_HQOS`,
`VTSS_DEBUG_GROUP_VXLAT`,
`VTSS_DEBUG_GROUP_OAM`, `VTSS_DEBUG_GROUP_SER_GPIO`, `VTSS_DEBUG_GROUP_L3`,
`VTSS_DEBUG_GROUP_AFI`,
`VTSS_DEBUG_GROUP_MACSEC`, `VTSS_DEBUG_GROUP_COUNT` }

Debug function group.

- enum `vtss_ptp_event_type_t`{
`VTSS_PTP_SYNC_EV` = `(1 << 0)`, `VTSS_PTP_EXT_SYNC_EV` = `(1 << 1)`, `VTSS_PTP_CLK_ADJ_EV` =
`(1 << 2)`, `VTSS_PTP_TX_TSTAMP_EV` = `(1 << 3)`,
`VTSS_PTP_EXT_1_SYNC_EV` = `(1 << 4)` }

Define event (interrupt) types relatesd to PTP in the switch chips.

- enum `vtss_dev_all_event_type_t`{ `VTSS_DEV_ALL_TX_TSTAMP_EV` = `(1 << 0)`, `VTSS_DEV_ALL_LINK_EV` =
`(1 << 1)` }

Define the dev_all event (interrupt) types.

- enum `vtss_dev_all_event_poll_t`{ `VTSS_DEV_ALL_POLL_ALL`, `VTSS_DEV_ALL_POLL_PRIMARY`, `VTSS_DEV_ALL_POLL_SECONDARY` }

Define the dev_all polling types.

- enum `vtss_gpio_mode_t`{
`VTSS_GPIO_OUT`, `VTSS_GPIO_IN`, `VTSS_GPIO_IN_INT`, `VTSS_GPIO_ALT_0`,
`VTSS_GPIO_ALT_1`, `VTSS_GPIO_ALT_2` }

- enum `vtss_sgpio_mode_t` {

VTSS_SGPIO_MODE_OFF, VTSS_SGPIO_MODE_ON, VTSS_SGPIO_MODE_0, VTSS_SGPIO_MODE_1,

VTSS_SGPIO_MODE_0_ACTIVITY, VTSS_SGPIO_MODE_1_ACTIVITY, VTSS_SGPIO_MODE_0_ACTIVITY_INV,

VTSS_SGPIO_MODE_1_ACTIVITY_INV }

SGPIO configured mode.
- enum `vtss_sgpio_bmode_t` {

VTSS_SGPIO_BMODE_TOGGLE, VTSS_SGPIO_BMODE_0_625, VTSS_SGPIO_BMODE_1_25,

VTSS_SGPIO_BMODE_2_5,

VTSS_SGPIO_BMODE_5 }

SGPIO output mode.
- enum `vtss_irq_t` {

VTSS_IRQ_XTR, VTSS_IRQ_FDMA_XTR, VTSS_IRQ_SOFTWARE, VTSS_IRQ_PTP_RDY,

VTSS_IRQ_PTP_SYNC, VTSS_IRQ_EXT1, VTSS_IRQ_OAM, VTSS_IRQ_MAX }

Interrupt sources.
- enum `vtss_fan_pwd_freq_t` {

VTSS_FAN_PWM_FREQ_25KHZ, VTSS_FAN_PWM_FREQ_120HZ, VTSS_FAN_PWM_FREQ_100HZ,

VTSS_FAN_PWM_FREQ_80HZ,

VTSS_FAN_PWM_FREQ_60HZ, VTSS_FAN_PWM_FREQ_40HZ, VTSS_FAN_PWM_FREQ_20HZ, VTSS_FAN_PWM_FREQ_10HZ }

FAN PWM frequency.
- enum `vtss_fan_type_t` { VTSS_FAN_2_WIRE_TYPE, VTSS_FAN_3_WIRE_TYPE, VTSS_FAN_4_WIRE_TYPE }

FAN Types.
- enum `vtss_eee_state_select_t` {

VTSS_EEE_STATE_SELECT_LPI, VTSS_EEE_STATE_SELECT_SCH, VTSS_EEE_STATE_SELECT_FP,

VTSS_EEE_STATE_SELECT_INTR_ENA,

VTSS_EEE_STATE_SELECT_INTR_ACK }

EEE port state change what? (JR only)

Functions

- `vtss_rc vtss_trace_conf_get (const vtss_trace_group_t group, vtss_trace_conf_t *const conf)`

Get trace configuration.
- `vtss_rc vtss_trace_conf_set (const vtss_trace_group_t group, const vtss_trace_conf_t *const conf)`

Set trace configuration.
- `void vtss_callout_trace_printf (const vtss_trace_layer_t layer, const vtss_trace_group_t group, const vtss_trace_level_t level, const char *file, const int line, const char *function, const char *format,...)`

Trace callout function.
- `void vtss_callout_trace_hex_dump (const vtss_trace_layer_t layer, const vtss_trace_group_t group, const vtss_trace_level_t level, const char *file, const int line, const char *function, const u8 *byte_p, const int byte_cnt)`

Trace hex-dump callout function.
- `vtss_rc vtss_debug_info_get (vtss_debug_info_t *const info)`

Get default debug information structure.
- `vtss_rc vtss_debug_info_print (const vtss_inst_t inst, const vtss_debug_printf_t printf, const vtss_debug_info_t *const info)`

Print default information.
- `void vtss_callout_lock (const vtss_api_lock_t *const lock)`

Lock API access.
- `void vtss_callout_unlock (const vtss_api_lock_t *const lock)`

Unlock API access.
- `vtss_rc vtss_debug_lock (const vtss_inst_t inst, const vtss_debug_lock_t *const lock)`

- Debug lock API access.*
- `vtss_rc vtss_debug_unlock` (const `vtss_inst_t` inst, `vtss_debug_lock_t` *const lock)

Debug unlock API access.
 - `vtss_rc vtss_reg_read` (const `vtss_inst_t` inst, const `vtss_chip_no_t` chip_no, const `u32` addr, `u32` *const value)

Read value from target register.
 - `vtss_rc vtss_reg_write` (const `vtss_inst_t` inst, const `vtss_chip_no_t` chip_no, const `u32` addr, const `u32` value)

Write value to target register.
 - `vtss_rc vtss_reg_write_masked` (const `vtss_inst_t` inst, const `vtss_chip_no_t` chip_no, const `u32` addr, const `u32` value, const `u32` mask)

Read, modify and write value to target register.
 - `vtss_rc vtss_intr_sticky_clear` (const `vtss_inst_t` inst, `u32` ext)

Clear EXT0-1 interrupt sticky bits on secondary chip.
 - `vtss_rc vtss_chip_id_get` (const `vtss_inst_t` inst, `vtss_chip_id_t` *const chip_id)

Get chip ID and revision.
 - `vtss_rc vtss_poll_1sec` (const `vtss_inst_t` inst)

Polling function called every second.
 - `vtss_rc vtss_ptp_event_poll` (const `vtss_inst_t` inst, `vtss_ptp_event_type_t` *const ev_mask)

PTP polling function called at by interrupt or periodically.
 - `vtss_rc vtss_ptp_event_enable` (const `vtss_inst_t` inst, const `vtss_ptp_event_type_t` ev_mask, const `BOOL` enable)

Enable PTP event generation for a specific event type.
 - `vtss_rc vtss_dev_all_event_poll` (const `vtss_inst_t` inst, const `vtss_dev_all_event_poll_t` poll_type, `vtss_dev_all_event_type_t` *const ev_mask)

DEV_ALL polling function called at by interrupt or periodically.
 - `vtss_rc vtss_dev_all_event_enable` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_dev_all_event_type_t` ev_mask, const `BOOL` enable)

Enable DEV_ALL event generation for a specific event type.
 - `vtss_rc vtss_gpio_mode_set` (const `vtss_inst_t` inst, const `vtss_chip_no_t` chip_no, const `vtss_gpio_no_t` gpio_no, const `vtss_gpio_mode_t` mode)

Set GPIO mode.
 - `vtss_rc vtss_gpio_direction_set` (const `vtss_inst_t` inst, const `vtss_chip_no_t` chip_no, const `vtss_gpio_no_t` gpio_no, const `BOOL` output)

Set GPIO direction to input or output.
 - `vtss_rc vtss_gpio_read` (const `vtss_inst_t` inst, const `vtss_chip_no_t` chip_no, const `vtss_gpio_no_t` gpio_no, `BOOL` *const value)

Read from GPIO input pin.
 - `vtss_rc vtss_gpio_write` (const `vtss_inst_t` inst, const `vtss_chip_no_t` chip_no, const `vtss_gpio_no_t` gpio_no, const `BOOL` value)

Write to GPIO output pin.
 - `vtss_rc vtss_gpio_event_poll` (const `vtss_inst_t` inst, const `vtss_chip_no_t` chip_no, `BOOL` *const events)

Get GPIO event indication.
 - `vtss_rc vtss_gpio_event_enable` (const `vtss_inst_t` inst, const `vtss_chip_no_t` chip_no, const `vtss_gpio_no_t` gpio_no, `BOOL` enable)

Set GPIO event enable.
 - `vtss_rc vtss_sgpi_conf_get` (const `vtss_inst_t` inst, const `vtss_chip_no_t` chip_no, const `vtss_sgpi_group_t` group, `vtss_sgpi_conf_t` *const conf)

Get SGPIO configuration.
 - `vtss_rc vtss_sgpi_conf_set` (const `vtss_inst_t` inst, const `vtss_chip_no_t` chip_no, const `vtss_sgpi_group_t` group, const `vtss_sgpi_conf_t` *const conf)

Set SGPIO configuration.
 - `vtss_rc vtss_sgpi_read` (const `vtss_inst_t` inst, const `vtss_chip_no_t` chip_no, const `vtss_sgpi_group_t` group, `vtss_sgpi_port_data_t` data[`VTSS_SGPIO_PORTS`])

Read SGPIO data.

- *Read GPIO data.*
- **vtss_rc vtss_sgpio_event_poll** (const **vtss_inst_t** inst, const **vtss_chip_no_t** chip_no, const **vtss_sgpio_group_t** group, const **u32** bit, **BOOL** *const events)
 - Get GPIO event indication.*
- **vtss_rc vtss_sgpio_event_enable** (const **vtss_inst_t** inst, const **vtss_chip_no_t** chip_no, const **vtss_sgpio_group_t** group, const **vtss_port_no_t** port, const **u32** bit, **BOOL** enable)
 - Get GPIO event enable.*
- **vtss_rc vtss_intr_cfg** (const **vtss_inst_t** inst, const **u32** mask, const **BOOL** polarity, const **BOOL** enable)
 - Configure interrupt.*
- **vtss_rc vtss_irq_conf_get** (const **vtss_inst_t** inst, const **vtss_irq_t** irq, **vtss_irq_conf_t** *conf)
 - Get IRQ configuration.*
- **vtss_rc vtss_irq_conf_set** (const **vtss_inst_t** inst, const **vtss_irq_t** irq, const **vtss_irq_conf_t** *const conf)
 - Set IRQ configuration.*
- **vtss_rc vtss_irq_status_get_and_mask** (const **vtss_inst_t** inst, **vtss_irq_status_t** *status)
 - Get IRQ status (active sources), mask current sources.*
- **vtss_rc vtss_irq_enable** (const **vtss_inst_t** inst, const **vtss_irq_t** irq, **BOOL** enable)
 - Control a specific interrupt source.*
- **u32 vtss_tod_get_ns_cnt** (**void**)
 - Get the current hw nanosec time This function is called from interrupt.*
- **void vtss_tod_set_ns_cnt_cb** (**tod_get_ns_cnt_cb_t** cb)
 - Set an external hw nanosec read function.*
- **vtss_rc vtss_temp_sensor_init** (const **vtss_inst_t** inst, const **BOOL** enable)
 - Initialize the temperature sensor.*
- **vtss_rc vtss_temp_sensor_get** (const **vtss_inst_t** inst, **i16** *temperature)
 - Read temperature sensor value.*
- **vtss_rc vtss_fan_rotation_get** (const **vtss_inst_t** inst, **vtss_fan_conf_t** *const fan_spec, **u32** *rotation_count)
 - Get the number of fan rotations.*
- **vtss_rc vtss_fan_cool_lvl_set** (const **vtss_inst_t** inst, **u8** lvl)
 - Set fan cool level (Duty cycle)*
- **vtss_rc vtss_fan_controller_init** (const **vtss_inst_t** inst, const **vtss_fan_conf_t** *const spec)
 - Initialise fan controller)*
- **vtss_rc vtss_fan_cool_lvl_get** (const **vtss_inst_t** inst, **u8** *lvl)
 - Get fan cool level (Duty cycle)*
- **vtss_rc vtss_eee_port_conf_set** (const **vtss_inst_t** inst, const **vtss_port_no_t** port_no, const **vtss_eee_port_conf_t** *const eee_conf)
 - Set EEE configuration.*
- **vtss_rc vtss_eee_port_state_set** (const **vtss_inst_t** inst, const **vtss_port_no_t** port_no, const **vtss_eee_port_state_t** *const eee_state)
 - Change EEE Port state.*
- **vtss_rc vtss_eee_port_counter_get** (const **vtss_inst_t** inst, const **vtss_port_no_t** port_no, **vtss_eee_port_counter_t** *const eee_counter)
 - Get EEE-related port counters.*
- **vtss_rc vtss_debug_reg_check_set** (const **vtss_inst_t** inst, const **BOOL** enable)
 - Enable or disable register access checking.*

8.19.1 Detailed Description

Miscellaneous API.

This header file describes miscellaneous API functions

8.19.2 Macro Definition Documentation

8.19.2.1 VTSS_OS_TIMESTAMP_TYPE

```
#define VTSS_OS_TIMESTAMP_TYPE vtss_os_timestamp_t
```

The VTSS_OS_TIME_STAMP_TYPE defines the type

Definition at line 1075 of file vtss_misc_api.h.

8.19.2.2 VTSS_OS_TIMESTAMP

```
#define VTSS_OS_TIMESTAMP( \
    timestamp )
```

Value:

```
do { \
    /* Currently no need to lock scheduler, since it's only */ \
    /* called from a function, where the sceduler is already locked. */ \
    /* cyg_scheduler_lock(__FILE__, __LINE__); */ \
    (timestamp)->hw_cnt = vtss_tod_get_ns_cnt(); \
    /* cyg_scheduler_unlock(__FILE__, __LINE__); */ \
} while(0);
```

`VTSS_OS_TIMESTAMP()` provides the implementation that will fill in the timestamp.

Definition at line 1076 of file vtss_misc_api.h.

8.19.3 Typedef Documentation

8.19.3.1 tod_get_ns_cnt_cb_t

```
typedef u32 (* tod_get_ns_cnt_cb_t) (void)
```

If the actual HW does not support time stamping, an external callback function can be set up to do the work.

Returns

actual ns counter.

Definition at line 1054 of file vtss_misc_api.h.

8.19.4 Enumeration Type Documentation

8.19.4.1 vtss_trace_layer_t

```
enum vtss_trace_layer_t
```

Trace group layer.

Enumerator

VTSS_TRACE_LAYER_AIL	Application Interface Layer
VTSS_TRACE_LAYER_CIL	Chip Interface Layer
VTSS_TRACE_LAYER_COUNT	Number of layers

Definition at line 43 of file vtss_misc_api.h.

8.19.4.2 vtss_trace_group_t

```
enum vtss_trace_group_t
```

Trace groups.

Enumerator

VTSS_TRACE_GROUP_DEFAULT	Default trace group
VTSS_TRACE_GROUP_PORT	Port control
VTSS_TRACE_GROUP_PHY	PHY control
VTSS_TRACE_GROUP_PACKET	Packet control
VTSS_TRACE_GROUP_AFI	AFI
VTSS_TRACE_GROUP_QOS	Quality of Service
VTSS_TRACE_GROUP_L2	Layer 2
VTSS_TRACE_GROUP_L3	Layer 3
VTSS_TRACE_GROUP_SECURITY	Security
VTSS_TRACE_GROUP_EVC	Ethernet Virtual Connections
VTSS_TRACE_GROUP_FDMA_NORMAL	Frame DMA Extraction and Injection when interrupts/scheduler is enabled
VTSS_TRACE_GROUP_FDMA_IRQ	Frame DMA when interrupts/scheduler is disabled Daytona layers are placed before the PHY layer, otherwise they are not shown in the CLI commands
VTSS_TRACE_GROUP_REG_CHECK	Register access errors (must be able to print when interrupts/scheduler is disabled)
VTSS_TRACE_GROUP MPLS	MPLS
VTSS_TRACE_GROUP_HQOS	Hierarchical Quality of Service
VTSS_TRACE_GROUP_MACSEC	MACSEC control
VTSS_TRACE_GROUP_VCAP	VCAP
VTSS_TRACE_GROUP_OAM	OAM
VTSS_TRACE_GROUP_TS	Timestamping
VTSS_TRACE_GROUP_COUNT	Number of trace groups

Definition at line 52 of file vtss_misc_api.h.

8.19.4.3 vtss_trace_level_t

```
enum vtss_trace_level_t
```

Trace levels.

Enumerator

VTSS_TRACE_LEVEL_NONE	No trace
VTSS_TRACE_LEVEL_ERROR	Error trace
VTSS_TRACE_LEVEL_INFO	Information trace
VTSS_TRACE_LEVEL_DEBUG	Debug trace
VTSS_TRACE_LEVEL_NOISE	More debug information
VTSS_TRACE_LEVEL_COUNT	Number of trace levels

Definition at line 85 of file vtss_misc_api.h.

8.19.4.4 vtss_debug_layer_t

```
enum vtss_debug_layer_t
```

Debug layer.

Enumerator

VTSS_DEBUG_LAYER_ALL	All layers
VTSS_DEBUG_LAYER_AIL	Application Interface Layer
VTSS_DEBUG_LAYER_CIL	Chip Interface Layer

Definition at line 177 of file vtss_misc_api.h.

8.19.4.5 vtss_debug_group_t

```
enum vtss_debug_group_t
```

Debug function group.

Enumerator

VTSS_DEBUG_GROUP_ALL	All groups
VTSS_DEBUG_GROUP_INIT	Initialization
VTSS_DEBUG_GROUP_MISC	Miscellaneous
VTSS_DEBUG_GROUP_PORT	Port configuration
VTSS_DEBUG_GROUP_PORT_CNT	Port counters
VTSS_DEBUG_GROUP_PHY	PHY
VTSS_DEBUG_GROUP_VLAN	VLAN
VTSS_DEBUG_GROUP_PVLAN	PVLAN
VTSS_DEBUG_GROUP_MAC_TABLE	MAC address table
VTSS_DEBUG_GROUP_ACL	ACL

Enumerator

VTSS_DEBUG_GROUP_QOS	QoS
VTSS_DEBUG_GROUP_AGGR	Link aggregation
VTSS_DEBUG_GROUP_GLAG	Global link aggregation
VTSS_DEBUG_GROUP_STP	Spanning Tree
VTSS_DEBUG_GROUP_MIRROR	Mirroring
VTSS_DEBUG_GROUP_EVC	EVC
VTSS_DEBUG_GROUP_ERPS	ERPS
VTSS_DEBUG_GROUP_EPS	EPS
VTSS_DEBUG_GROUP_PACKET	Packet control
VTSS_DEBUG_GROUP_FDMA	FDMA
VTSS_DEBUG_GROUP_TS	TS: TimeStamping
VTSS_DEBUG_GROUP_PHY_TS	PHY_TS: PHY TimeStamping
VTSS_DEBUG_GROUP_WM	WaterMarks
VTSS_DEBUG_GROUP_LRN	LRN:COMMON
VTSS_DEBUG_GROUP_IPMC	IP Multicast
VTSS_DEBUG_GROUP_STACK	Stacking
VTSS_DEBUG_GROUP_CMEF	Congestion Management
VTSS_DEBUG_GROUP_HOST	CE-MAX Host configuration
VTSS_DEBUG_GROUP MPLS	MPLS
VTSS_DEBUG_GROUP_MPLS_OAM	MPLS OAM
VTSS_DEBUG_GROUP_HQOS	Hierarchical Quality of Service
VTSS_DEBUG_GROUP_VXLAT	VLAN Translation
VTSS_DEBUG_GROUP_OAM	OAM, incl. VOEs/VOP
VTSS_DEBUG_GROUP_SER_GPIO	Serial GPIO configuration
VTSS_DEBUG_GROUP_L3	L3 services
VTSS_DEBUG_GROUP_AFI	Automatic Frame Injector
VTSS_DEBUG_GROUP_MACSEC	802.1AE MacSec
VTSS_DEBUG_GROUP_COUNT	Number of groups

Definition at line 184 of file vtss_misc_api.h.

8.19.4.6 vtss_gpio_mode_t

```
enum vtss_gpio_mode_t
```

GPIO configured mode.

Enumerator

VTSS_GPIO_OUT	Output enabled
VTSS_GPIO_IN	Input enabled
VTSS_GPIO_IN_INT	Input enabled, IRQ gated
VTSS_GPIO_ALT_0	Alternate function 0
VTSS_GPIO_ALT_1	Alternate function 1
	Alternate function 2
Generated by OpenOCD	OpenALT_2

Definition at line 567 of file vtss_misc_api.h.

8.19.4.7 vtss_sgpiemode_t

enum `vtss_sgpiemode_t`

SGPIO output mode.

Enumerator

<code>VTSS_SGPIO_MODE_OFF</code>	Off
<code>VTSS_SGPIO_MODE_ON</code>	On
<code>VTSS_SGPIO_MODE_0</code>	Mode 0
<code>VTSS_SGPIO_MODE_1</code>	Mode 1
<code>VTSS_SGPIO_MODE_0_ACTIVITY</code>	Mode 0 when link activity
<code>VTSS_SGPIO_MODE_1_ACTIVITY</code>	Mode 1 when link activity
<code>VTSS_SGPIO_MODE_0_ACTIVITY_INV</code>	Mode 0 when link activity, inversed polarity
<code>VTSS_SGPIO_MODE_1_ACTIVITY_INV</code>	Mode 1 when link activity, inversed polarity

Definition at line 766 of file vtss_misc_api.h.

8.19.4.8 vtss_sgpiobmode_t

enum `vtss_sgpiobmode_t`

SGPIO blink mode.

Enumerator

<code>VTSS_SGPIO_BMODE_TOGGLE</code>	Burst toggle (mode 1 only)
<code>VTSS_SGPIO_BMODE_0_625</code>	0.625 Hz (mode 0 only)
<code>VTSS_SGPIO_BMODE_1_25</code>	1.25 Hz
<code>VTSS_SGPIO_BMODE_2_5</code>	2.5 Hz
<code>VTSS_SGPIO_BMODE_5</code>	5 Hz

Definition at line 779 of file vtss_misc_api.h.

8.19.4.9 vtss_irq_t

enum `vtss_irq_t`

Interrupt sources.

Enumerator

VTSS_IRQ_XTR	Frame Extraction Ready(register-based)
VTSS_IRQ_FDMA_XTR	Frame Extraction Ready (FDMA-based)
VTSS_IRQ_SOFTWARE	Software IRQ
VTSS_IRQ_PTP_RDY	PTP Timestamp Ready
VTSS_IRQ_PTP_SYNC	PTP Synchronization IRQ
VTSS_IRQ_EXT1	EXT1 IRQ
VTSS_IRQ_OAM	OAM IRQ
VTSS_IRQ_MAX	Maximum IRQ Source

Definition at line 957 of file vtss_misc_api.h.

8.19.4.10 vtss_eee_state_select_t

```
enum vtss_eee_state_select_t
```

EEE port state change what? (JR only)

Enumerator

VTSS_EEE_STATE_SELECT_LPI	Change LPI signal.
VTSS_EEE_STATE_SELECT_SCH	Change scheduler enable.
VTSS_EEE_STATE_SELECT_FP	Change frame mirroring flag.
VTSS_EEE_STATE_SELECT_INTR_ENA	Enable analyzer interrupts.
VTSS_EEE_STATE_SELECT_INTR_ACK	Acknowledge analyzer interrupts.

Definition at line 1230 of file vtss_misc_api.h.

8.19.5 Function Documentation

8.19.5.1 vtss_trace_conf_get()

```
vtss_rc vtss_trace_conf_get (
    const vtss_trace_group_t group,
    vtss_trace_conf_t *const conf )
```

Get trace configuration.

Parameters

group	[IN] Trace group
conf	[OUT] Trace group configuration.

Returns

Return code.

8.19.5.2 vtss_trace_conf_set()

```
vtss_rc vtss_trace_conf_set (
    const vtss_trace_group_t group,
    const vtss_trace_conf_t *const conf )
```

Set trace configuration.

Parameters

<i>group</i>	[IN] Trace group
<i>conf</i>	[IN] Trace group configuration.

Returns

Return code.

8.19.5.3 vtss_callout_trace_printf()

```
void vtss_callout_trace_printf (
    const vtss_trace_layer_t layer,
    const vtss_trace_group_t group,
    const vtss_trace_level_t level,
    const char * file,
    const int line,
    const char * function,
    const char * format,
    ... )
```

Trace callout function.

Parameters

<i>layer</i>	[IN] Trace layer
<i>group</i>	[IN] Trace group
<i>level</i>	[IN] Trace level
<i>file</i>	[IN] File name string
<i>line</i>	[IN] Line number in file
<i>function</i>	[IN] Function name string
<i>format</i>	[IN] Print format string

Returns

Nothing.

8.19.5.4 vtss_callout_trace_hex_dump()

```
void vtss_callout_trace_hex_dump (
    const vtss_trace_layer_t layer,
    const vtss_trace_group_t group,
    const vtss_trace_level_t level,
    const char * file,
    const int line,
    const char * function,
    const u8 * byte_p,
    const int byte_cnt )
```

Trace hex-dump callout function.

Parameters

<i>layer</i>	[IN] Trace layer
<i>group</i>	[IN] Trace group
<i>level</i>	[IN] Trace level
<i>file</i>	[IN] The file from where the trace were called.
<i>line</i>	[IN] The line from where the trace were called.
<i>function</i>	[IN] The function from where the trace were called.
<i>byte_p</i>	[IN] Pointer to start of area to print
<i>byte_cnt</i>	[IN] Number of bytes to print

Returns

Nothing.

8.19.5.5 vtss_debug_info_get()

```
vtss_rc vtss_debug_info_get (
    vtss_debug_info_t *const info )
```

Get default debug information structure.

Parameters

<i>info</i>	[OUT] Debug information
-------------	-------------------------

Returns

Return code.

8.19.5.6 vtss_debug_info_print()

```
vtss_rc vtss_debug_info_print (
    const vtss_inst_t inst,
    const vtss_debug_printf_t prntf,
    const vtss_debug_info_t *const info )
```

Print default information.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>prntf</i>	[IN] Debug printf function.
<i>info</i>	[IN] Debug information

Returns

Return code.

8.19.5.7 vtss_callout_lock()

```
void vtss_callout_lock (
    const vtss_api_lock_t *const lock )
```

Lock API access.

Parameters

<i>lock</i>	[IN] Lock information
-------------	-----------------------

8.19.5.8 vtss_callout_unlock()

```
void vtss_callout_unlock (
    const vtss_api_lock_t *const lock )
```

Unlock API access.

Parameters

<i>lock</i>	[IN] Lock information
-------------	-----------------------

8.19.5.9 vtss_debug_lock()

```
vtss_rc vtss_debug_lock (
    const vtss_inst_t inst,
    const vtss_debug_lock_t *const lock )
```

Debug lock API access.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>lock</i>	[IN] Lock information

Returns

Return code.

8.19.5.10 vtss_debug_unlock()

```
vtss_rc vtss_debug_unlock (
    const vtss_inst_t inst,
    vtss_debug_lock_t *const lock )
```

Debug unlock API access.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>lock</i>	[IN] Lock information

Returns

Return code.

8.19.5.11 vtss_reg_read()

```
vtss_rc vtss_reg_read (
    const vtss_inst_t inst,
```

```
const vtss_chip_no_t chip_no,
const u32 addr,
u32 *const value )
```

Read value from target register.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>chip_no</i>	[IN] Chip number (if multi-chip instance).
<i>addr</i>	[IN] Address to read. Format depends on target.
<i>value</i>	[OUT] Register value.

Returns

Return code.

8.19.5.12 vtss_reg_write()

```
vtss_rc vtss_reg_write (
    const vtss_inst_t inst,
    const vtss_chip_no_t chip_no,
    const u32 addr,
    const u32 value )
```

Write value to target register.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>chip_no</i>	[IN] Chip number (if multi-chip instance).
<i>addr</i>	[IN] Address to read. Format depends on target.
<i>value</i>	[IN] Register value.

Returns

Return code.

8.19.5.13 vtss_reg_write_masked()

```
vtss_rc vtss_reg_write_masked (
    const vtss_inst_t inst,
    const vtss_chip_no_t chip_no,
    const u32 addr,
    const u32 value,
    const u32 mask )
```

Read, modify and write value to target register.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>chip_no</i>	[IN] Chip number (if multi-chip instance).
<i>addr</i>	[IN] Address to read. Format depends on target.
<i>value</i>	[IN] Register value.
<i>mask</i>	[IN] Register mask, only bits enabled are changed.

Returns

Return code.

8.19.5.14 vtss_intr_sticky_clear()

```
vtss_rc vtss_intr_sticky_clear (
    const vtss_inst_t inst,
    u32 ext )
```

Clear EXT0-1 interrupt sticky bits on secondary chip.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>ext</i>	[IN] EXT number (0-1).

Returns

Return code.

8.19.5.15 vtss_chip_id_get()

```
vtss_rc vtss_chip_id_get (
    const vtss_inst_t inst,
    vtss_chip_id_t *const chip_id )
```

Get chip ID and revision.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>chip_id</i>	[IN] Pointer to chip ID structure.

Returns

Return code.

8.19.5.16 vtss_poll_1sec()

```
vtss_rc vtss_poll_1sec (
    const vtss_inst_t inst )
```

Polling function called every second.

Parameters

<i>inst</i>	[IN] Target instance reference.
-------------	---------------------------------

Returns

Return code.

8.19.5.17 vtss_ptp_event_poll()

```
vtss_rc vtss_ptp_event_poll (
    const vtss_inst_t inst,
    vtss_ptp_event_type_t *const ev_mask )
```

PTP polling function called at by interrupt or periodically.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>ev_mask</i>	[OUT] Event type mask of active events

Note

The *ev_mask* parameter can be either a single event_type or multiple event types (or *VTSS_EVTYPE_ALL*). If invoked by a processor interrupt signal, the type of event to check for may be narrowed in to specific events.

Returns

Return code.

8.19.5.18 vtss_ptp_event_enable()

```
vtss_rc vtss_ptp_event_enable (
    const vtss_inst_t inst,
    const vtss_ptp_event_type_t ev_mask,
    const BOOL enable )
```

Enable PTP event generation for a specific event type.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>enable</i>	[IN] Enable or disable events
<i>ev_mask</i>	[IN] Event type(s) to control (mask)

Returns

Return code.

8.19.5.19 vtss_dev_all_event_poll()

```
vtss_rc vtss_dev_all_event_poll (
    const vtss_inst_t inst,
    const vtss_dev_all_event_poll_t poll_type,
    vtss_dev_all_event_type_t *const ev_mask )
```

DEV_ALL polling function called at by interrupt or periodically.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>poll_type</i>	[IN] Polling type
<i>ev_mask</i>	[OUT] Event type mask array of active events for all ports - must be of size VTSS_PORT_ARRAY_SIZE

Note

The *ev_mask* parameter can be either a single event_type or multiple event types (or VTSS_EVTYPE_ALL). If invoked by a processor interrupt signal, the type of event to check for may be narrowed in to specific events.

Returns

Return code.

8.19.5.20 vtss_dev_all_event_enable()

```
vtss_rc vtss_dev_all_event_enable (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_dev_all_event_type_t ev_mask,
    const BOOL enable )
```

Enable DEV_ALL event generation for a specific event type.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>enable</i>	[IN] Enable or disable events.
<i>ev_mask</i>	[IN] Event type(s) to control (mask).

Returns

Return code.

8.19.5.21 vtss_gpio_mode_set()

```
vtss_rc vtss_gpio_mode_set (
    const vtss_inst_t inst,
    const vtss_chip_no_t chip_no,
    const vtss_gpio_no_t gpio_no,
    const vtss_gpio_mode_t mode )
```

Set GPIO mode.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>chip_no</i>	[IN] Chip number (if multi-chip instance).
<i>gpio_no</i>	[IN] GPIO pin number.
<i>mode</i>	[IN] GPIO mode.

Returns

Return code.

8.19.5.22 vtss_gpio_direction_set()

```
vtss_rc vtss_gpio_direction_set (
    const vtss_inst_t inst,
```

```
const vtss_chip_no_t chip_no,
const vtss_gpio_no_t gpio_no,
const BOOL output )
```

Set GPIO direction to input or output.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>chip_no</i>	[IN] Chip number (if multi-chip instance).
<i>gpio_no</i>	[IN] GPIO pin number.
<i>output</i>	[IN] TRUE if output, FALSE if input.

Returns

Return code.

DEPRECATED. Use [vtss_gpio_mode_set\(\)](#) instead.

8.19.5.23 vtss_gpio_read()

```
vtss_rc vtss_gpio_read (
    const vtss_inst_t inst,
    const vtss_chip_no_t chip_no,
    const vtss_gpio_no_t gpio_no,
    BOOL *const value )
```

Read from GPIO input pin.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>chip_no</i>	[IN] Chip number (if multi-chip instance).
<i>gpio_no</i>	[IN] GPIO pin number.
<i>value</i>	[OUT] TRUE if pin is high, FALSE if it is low.

Returns

Return code.

8.19.5.24 vtss_gpio_write()

```
vtss_rc vtss_gpio_write (
    const vtss_inst_t inst,
    const vtss_chip_no_t chip_no,
    const vtss_gpio_no_t gpio_no,
    const BOOL value )
```

Write to GPIO output pin.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>chip_no</i>	[IN] Chip number (if multi-chip instance).
<i>gpio_no</i>	[IN] GPIO pin number.
<i>value</i>	[IN] TRUE to set pin high, FALSE to set pin low.

Returns

Return code.

8.19.5.25 vtss_gpio_event_poll()

```
vtss_rc vtss_gpio_event_poll (
    const vtss_inst_t inst,
    const vtss_chip_no_t chip_no,
    BOOL *const events )
```

Get GPIO event indication.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>chip_no</i>	[IN] Chip number (if multi-chip instance).
<i>events</i>	[OUT] Event indication for each GPIO pin - must point to VTSS_GPIOS of BOOL.

Returns

Return code.

8.19.5.26 vtss_gpio_event_enable()

```
vtss_rc vtss_gpio_event_enable (
    const vtss_inst_t inst,
    const vtss_chip_no_t chip_no,
    const vtss_gpio_no_t gpio_no,
    BOOL enable )
```

Set GPIO event enable.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>chip_no</i>	[IN] Chip number (if multi-chip instance).
<i>gpio_no</i>	[IN] GPIO pin number.
<i>enable</i>	[IN] Enable or disable event.

Returns

Return code.

8.19.5.27 vtss_sgpio_conf_get()

```
vtss_rc vtss_sgpio_conf_get (
    const vtss_inst_t inst,
    const vtss_chip_no_t chip_no,
    const vtss_sgpio_group_t group,
    vtss_sgpio_conf_t *const conf )
```

Get GPIO configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>chip_no</i>	[IN] Chip number (if multi-chip instance).
<i>group</i>	[IN] GPIO group.
<i>conf</i>	[OUT] GPIO configuration.

Returns

Return code.

8.19.5.28 vtss_sgpio_conf_set()

```
vtss_rc vtss_sgpio_conf_set (
    const vtss_inst_t inst,
    const vtss_chip_no_t chip_no,
    const vtss_sgpio_group_t group,
    const vtss_sgpio_conf_t *const conf )
```

Set GPIO configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>chip_no</i>	[IN] Chip number (if multi-chip instance).
<i>group</i>	[IN] GPIO group.
<i>conf</i>	[IN] GPIO configuration.

Returns

Return code.

8.19.5.29 vtss_sgpio_read()

```
vtss_rc vtss_sgpio_read (
    const vtss_inst_t inst,
    const vtss_chip_no_t chip_no,
    const vtss_sgpio_group_t group,
    vtss_sgpio_port_data_t data[VTSS_SGPIO_PORTS] )
```

Read GPIO data.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>chip_no</i>	[IN] Chip number (if multi-chip instance).
<i>group</i>	[IN] GPIO group.
<i>data</i>	[OUT] GPIO data.

Returns

Return code.

8.19.5.30 vtss_sgpio_event_poll()

```
vtss_rc vtss_sgpio_event_poll (
    const vtss_inst_t inst,
    const vtss_chip_no_t chip_no,
    const vtss_sgpio_group_t group,
    const u32 bit,
    BOOL *const events )
```

Get GPIO event indication.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>chip_no</i>	[IN] Chip number (if multi-chip instance).
<i>group</i>	[IN] GPIO group.
<i>bit</i>	[IN] GPIO port bit (0-3).
<i>events</i>	[OUT] Event indication for each port for the selected bit - must point to VTSS_SGPIO_PORTS of BOOL.

Returns

Return code.

8.19.5.31 vtss_sgpio_event_enable()

```
vtss_rc vtss_sgpio_event_enable (
    const vtss_inst_t inst,
    const vtss_chip_no_t chip_no,
    const vtss_sgpio_group_t group,
    const vtss_port_no_t port,
    const u32 bit,
    BOOL enable )
```

Get GPIO event enable.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>chip_no</i>	[IN] Chip number (if multi-chip instance).
<i>group</i>	[IN] GPIO group.
<i>port</i>	[IN] GPIO port (0-31).
<i>bit</i>	[IN] GPIO port bit (0-3).
<i>enable</i>	[IN] Event for each port for the selected bit is enabled or disabled.

Returns

Return code.

8.19.5.32 vtss_intr_cfg()

```
vtss_rc vtss_intr_cfg (
    const vtss_inst_t inst,
    const u32 mask,
    const BOOL polarity,
    const BOOL enable )
```

Configure interrupt.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>mask</i>	[IN] Interrupt mask - Configures the interrupts for the bits set in the mask.
<i>polarity</i>	[IN] Polarity - Interrupt polarity.
<i>enable</i>	[IN] Enable - 1 = enable, 0 = disable.

Returns

Return code.

8.19.5.33 vtss_irq_conf_get()

```
vtss_rc vtss_irq_conf_get (
    const vtss_inst_t inst,
    const vtss_irq_t irq,
    vtss_irq_conf_t * conf )
```

Get IRQ configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>irq</i>	[IN] Interrupt source.
<i>conf</i>	[OUT] IRQ configuration.

Returns

Return code.

8.19.5.34 vtss_irq_conf_set()

```
vtss_rc vtss_irq_conf_set (
    const vtss_inst_t inst,
    const vtss_irq_t irq,
    const vtss_irq_conf_t *const conf )
```

Set IRQ configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>irq</i>	[IN] Interrupt source.
<i>conf</i>	[IN] IRQ configuration.

Returns

Return code.

8.19.5.35 vtss_irq_status_get_and_mask()

```
vtss_rc vtss_irq_status_get_and_mask (
    const vtss_inst_t inst,
    vtss_irq_status_t * status )
```

Get IRQ status (active sources), mask current sources.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>status</i>	[OUT] IRQ status.

Returns

Return code.

8.19.5.36 vtss_irq_enable()

```
vtss_rc vtss_irq_enable (
    const vtss_inst_t inst,
    const vtss_irq_t irq,
    BOOL enable )
```

Control a specific interrupt source.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>irq</i>	[IN] Interrupt source.
<i>enable</i>	[IN] Enable or disable source.

Returns

Return code.

8.19.5.37 vtss_tod_get_ns_cnt()

```
u32 vtss_tod_get_ns_cnt (
    void )
```

Get the current hw nanosec time This function is called from interrupt.

Returns

actual ns counter

8.19.5.38 vtss_tod_set_ns_cnt_cb()

```
void vtss_tod_set_ns_cnt_cb (
    tod_get_ns_cnt_cb_t cb )
```

Set an external hw nanosec read function.

Parameters

<code>cb</code>	pointer to callback function
-----------------	------------------------------

8.19.5.39 vtss_temp_sensor_init()

```
vtss_rc vtss_temp_sensor_init (
    const vtss_inst_t inst,
    const BOOL enable )
```

Initialize the temperature sensor.

Parameters

<code>inst</code>	[IN] Target instance reference
<code>enable</code>	[IN] Set to true if sensor shall be active else false

Returns

Return code.

8.19.5.40 vtss_temp_sensor_get()

```
vtss_rc vtss_temp_sensor_get (
    const vtss_inst_t inst,
    i16 * temperature )
```

Read temperature sensor value.

Parameters

<code>inst</code>	[IN] Target instance reference
<code>temperature</code>	[OUT] Temperature from sensor (range from -46 to 135 degC)

Returns

Return code.

8.19.5.41 vtss_fan_rotation_get()

```
vtss_rc vtss_fan_rotation_get (
    const vtss_inst_t inst,
```

```
vtss_fan_conf_t *const fan_spec,
u32 * rotation_count )
```

Get the number of fan rotations.

Parameters

<i>inst</i>	[IN] Target instance reference
<i>fan_spec</i>	[IN] Fan specification
<i>rotation_count</i>	[OUT] Number of fan rotation countered for the last second.

Returns

Return code.

8.19.5.42 vtss_fan_cool_lvl_set()

```
vtss_rc vtss_fan_cool_lvl_set (
    const vtss_inst_t inst,
    u8 lvl )
```

Set fan cool level (Duty cycle)

Parameters

<i>inst</i>	[IN] Target instance reference
<i>lvl</i>	[IN] Level. 0 = Fan off, 255 = fan fully on

Returns

Return code.

8.19.5.43 vtss_fan_controller_init()

```
vtss_rc vtss_fan_controller_init (
    const vtss_inst_t inst,
    const vtss_fan_conf_t *const spec )
```

Initialise fan controller)

Parameters

<i>inst</i>	[IN] Target instance reference
<i>spec</i>	[IN] Fan specifications

Returns

Return code.

8.19.5.44 vtss_fan_cool_lvl_get()

```
vtss_rc vtss_fan_cool_lvl_get (
    const vtss_inst_t inst,
    u8 * lvl )
```

Get fan cool level (Duty cycle)

Parameters

<i>inst</i>	[IN] Target instance reference
<i>lvl</i>	[IN] Level. 0 = Fan off, 255 = fan fully on

Returns

Return code.

8.19.5.45 vtss_eee_port_conf_set()

```
vtss_rc vtss_eee_port_conf_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_eee_port_conf_t *const eee_conf )
```

Set EEE configuration.

Parameters

<i>inst</i>	[IN] Target instance reference
<i>port_no</i>	[IN] Port number
<i>eee_conf</i>	[IN] EEE configuration

Returns

Return code.

8.19.5.46 vtss_eee_port_state_set()

```
vtss_rc vtss_eee_port_state_set (
    const vtss_inst_t inst,
```

```
const vtss_port_no_t port_no,
const vtss_eee_port_state_t *const eee_state )
```

Change EEE Port state.

Supported on JR only.

Parameters

<i>inst</i>	[IN] Target instance reference
<i>port_no</i>	[IN] Port number
<i>eee_state</i>	[IN] New port state

Returns

Return code.

8.19.5.47 vtss_eee_port_counter_get()

```
vtss_rc vtss_eee_port_counter_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_eee_port_counter_t *const eee_counter )
```

Get EEE-related port counters.

Support on JR only.

Parameters

<i>inst</i>	[IN] Target instance reference
<i>port_no</i>	[IN] Port number
<i>eee_counter</i>	[INOUT] Structure indicating which counters to get, and the returned counter value.

Returns

Return code.

8.19.5.48 vtss_debug_reg_check_set()

```
vtss_rc vtss_debug_reg_check_set (
    const vtss_inst_t inst,
    const BOOL enable )
```

Enable or disable register access checking.

When enabled, every call to the register read/write callouts (`init_conf.reg_read()`/`write()`) will be verified for success. This will slow-down execution, so it is recommended only to use this feature in dedicated debug builds.

In multi-chip targets, only chip number 0 will be verified.

The number of calls will be reference counted, according to the following rules: 1) Calls with `enable = FALSE` will increase the reference count. 2) Calls with `enable = TRUE` will decrease the reference count, which cannot go lower than 0. The reference count is initialized to 1 at API instantiation, effectively disabling register access checking. A reference count of 0 enables the feature.

Error indications get printed with a call to `VTSS_EG(VTSS_TRACE_GROUP_REG_CHECK, ...)`, which will request to be interrupt/scheduler disabled tolerant.

Notice that this feature may not be available on all platforms.

This feature will not work if the API is instantiated more than once, since it uses the default instance to find its state.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>enable</i>	[IN] Enable or disable register access checking (ref. counted).

Returns

Return code.

8.20 vtss_api/include/vtss_mpls_api.h File Reference

MPLS API.

```
#include <vtss/api/types.h>
```

8.20.1 Detailed Description

MPLS API.

This header file describes the MPLS functions

8.21 vtss_api/include/vtss_oam_api.h File Reference

OAM API.

```
#include <vtss/api/types.h>
```

8.21.1 Detailed Description

OAM API.

This header file describes Y.1731/IEEE802.1ag OAM functions.

8.22 vtss_api/include/vtss_oha_api.h File Reference

OHA API.

```
#include <vtss/api/types.h>
```

8.22.1 Detailed Description

OHA API.

8.23 vtss_api/include/vtss_os.h File Reference

OS Layer API.

```
#include <vtss_os_linux.h>
```

8.23.1 Detailed Description

OS Layer API.

This header file includes the OS specific header file

8.24 vtss_api/include/vtss_os_custom.h File Reference

OS custom header file.

```
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
```

Macros

- #define `uint` unsigned int
- #define `ulong` unsigned long
- #define `VTSS_MSLEEP`(msec) <your function>
- #define `VTSS_MTIMER_START`(pTimer, msec) <your impl>
- #define `VTSS_MTIMER_TIMEOUT`(pTimer) <your impl>
- #define `VTSS_MTIMER_CANCEL`(pTimer) <your impl>
- #define `VTSS_DIV64`(dividend, divisor) <your impl>
- #define `VTSS_MOD64`(dividend, divisor) <your impl>
- #define `VTSS_LABS`(arg) <your impl>
- #define `VTSS_LLABS`(arg) <your impl>
- #define `VTSS_OS_CTZ`(val32) <your impl>
- #define `VTSS_OS_CTZ64`(val64) <your impl>
- #define `VTSS_OS_MALLOC`(size, flags) <your impl>
- #define `VTSS_OS_FREE`(ptr, flags) <your impl>
- #define `VTSS_OS_RAND`() <your impl>

Typedefs

- `typedef int vtss_mtimer_t`

8.24.1 Detailed Description

OS custom header file.

This file is a skeleton to be replaced by for customer specific OS.

8.24.2 Macro Definition Documentation

8.24.2.1 `uint`

```
#define uint unsigned int
```

Define API uint type - unsigned 16 bits

Definition at line 36 of file vtss_os_custom.h.

8.24.2.2 `ulong`

```
#define ulong unsigned long
```

Define API ulong - unsigned 32 bits

Definition at line 37 of file vtss_os_custom.h.

8.24.2.3 VTSS_MSLEEP

```
#define VTSS_MSLEEP(  
    msec ) <your function>
```

Your function for sleeping for "msec" milli seconds

Definition at line 40 of file vtss_os_custom.h.

8.24.2.4 VTSS_MTIMER_START

```
#define VTSS_MTIMER_START(  
    pTimer,  
    msec ) <your impl>
```

Start the timer (pTimer) with a timeout after a number of milliseconds

Definition at line 44 of file vtss_os_custom.h.

8.24.2.5 VTSS_MTIMER_TIMEOUT

```
#define VTSS_MTIMER_TIMEOUT(  
    pTimer ) <your impl>
```

Check if timer has timed out (Return TRUE in case of timeout else FALSE).

Definition at line 45 of file vtss_os_custom.h.

8.24.2.6 VTSS_MTIMER_CANCEL

```
#define VTSS_MTIMER_CANCEL(  
    pTimer ) <your impl>
```

Stop the timer

Definition at line 46 of file vtss_os_custom.h.

8.24.2.7 VTSS_DIV64

```
#define VTSS_DIV64(  
    dividend,  
    divisor ) <your impl>
```

support for 64 bit division

Definition at line 48 of file vtss_os_custom.h.

8.24.2.8 VTSS_MOD64

```
#define VTSS_MOD64 (
    dividend,
    divisor ) <your impl>
```

support for 64 bit division

Definition at line 49 of file vtss_os_custom.h.

8.24.2.9 VTSS_LABS

```
#define VTSS_LABS (
    arg ) <your impl>
```

long to abs

Definition at line 50 of file vtss_os_custom.h.

8.24.2.10 VTSS_LLABS

```
#define VTSS_LLABS (
    arg ) <your impl>
```

long long to abs

Definition at line 51 of file vtss_os_custom.h.

8.24.2.11 VTSS_OS_CTZ

```
#define VTSS_OS_CTZ (
    val32 ) <your impl>
```

Count trailing zeros of a 32-bit unsigned. Requirements/examples: `VTSS_OS_CTZ(0x00000001) = 0` `VTSS_OS_CTZ(0x80000000) = 31` `VTSS_OS_CTZ(0x00000000) >= 32` (if result is taken as unsigned; Most implementations return -1, and `(u32)(-1) >= 32`).

`__builtin_ctz()` is included in GCC 3.2.2 and later according to http://en.wikipedia.org/wiki/Find-first_set.

Definition at line 62 of file vtss_os_custom.h.

8.24.2.12 VTSS_OS_C TZ64

```
#define VTSS_OS_C TZ64( val64 ) <your impl>
```

Count trailing zeros of a 64-bit unsigned. Requirements/examples: `VTSS_OS_C TZ64(0x00000000_00000001) = 0` `VTSS_OS_C TZ64(0x00000000_80000000) = 31` `VTSS_OS_C TZ64(0x00000001_00000000) = 32` `VTSS_OS_C TZ64(0x80000000_00000000) = 63` `VTSS_OS_C TZ64(0x00000000_00000000) >= 64` (if result is taken as unsigned; Most implementations return -1, and `(u32)(-1) >= 64`).

Definition at line 73 of file vtss_os_custom.h.

8.24.2.13 VTSS_OS_MALLOC

```
#define VTSS_OS_MALLOC( size, flags ) <your impl>
```

Request OS to allocate size bytes of memory.

The first argument is the number of bytes that must be allocated. Type is `size_t`.

The second argument is a mask of flags that the implementation must obey. Type is `vtss_mem_flags_t`.

The returned pointer should be at least 8-byte aligned, to make it suitable for a struct.

Definition at line 87 of file vtss_os_custom.h.

8.24.2.14 VTSS_OS_FREE

```
#define VTSS_OS_FREE( ptr, flags ) <your impl>
```

Request OS to free memory previously allocated with `VTSS_OS_MALLOC()`.

The first argument is the pointer previously obtained with a call to `VTSS_OS_MALLOC()`. Type is `void *`.

The second argument is a mask of flags identical to those passed to `VTSS_OS_MALLOC()` when the memory was requested.

Definition at line 101 of file vtss_os_custom.h.

8.24.2.15 VTSS_OS_RAND

```
#define VTSS_OS_RAND( ) <your impl>
```

Wrap of call to `rand()` defined in `stdlib.h`

Definition at line 106 of file vtss_os_custom.h.

8.24.3 Typedef Documentation

8.24.3.1 vtss_mtimer_t

```
typedef int vtss_mtimer_t
```

Timer

Definition at line 43 of file vtss_os_custom.h.

8.25 vtss_api/include/vtss_os_ecos.h File Reference

eCos OS API

```
#include "vtss/api/types.h"
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <sys/types.h>
#include <cyg/kernel/kapi.h>
#include <cyg/hal/hal_cache.h>
#include <cyg/hal/hal_arch.h>
#include <cyg/hal/hal_endian.h>
```

Data Structures

- struct [vtss_timeofday_t](#)
Time of day structure.

Macros

- #define [VTSS_MSLEEP](#)(msec) HAL_DELAY_US(msec*1000)
- #define [VTSS_NSLEEP](#)(nsec) HAL_DELAY_US((nsec)/1000)
- #define [VTSS_MTIMER_START](#)(pTimer, msec) *pTimer = cyg_current_time() + ((msec)/10) + 1
- #define [VTSS_MTIMER_TIMEOUT](#)(pTimer) (cyg_current_time() > *(pTimer))
- #define [VTSS_MTIMER_CANCEL](#)(pTimer)
- #define [VTSS_TIME_OF_DAY](#)(tod) (tod.sec = (cyg_current_time() / CYGNUM_HAL_RTC_DENOMINATOR))
- #define [VTSS_DIV64](#)(dividend, divisor) ((dividend) / (divisor))
- #define [VTSS_MOD64](#)(dividend, divisor) ((dividend) % (divisor))
- #define [VTSS_LABS](#)(arg) labs(arg)
- #define [VTSS_LLabs](#)(arg) llabs(arg)
- #define [VTSS_OS_C TZ](#)(val32) ((val32) == 0 ? 32 : __builtin_ctz(val32))
- #define [VTSS_OS_C TZ64](#)(val64) ((val64) == 0 ? 64 : __builtin_ctzll(val64))
- #define [VTSS_OS_MALLOC](#)(size, flags) [vtss_callout_malloc](#)(size, flags)
- #define [VTSS_OS_FREE](#)(ptr, flags) [vtss_callout_free](#)(ptr, flags)
- #define [VTSS_OS_RAND](#)() rand()

- #define VTSS_OS_reordered_barrier() HAL_reordered_barrier()
 - #define VTSS_OS_COMPILER_ATTRIBUTE_ALIGNED(x) __attribute__((aligned(x)))
 - #define VTSS_OS_DCACHE_LINE_SIZE_BYTES HAL_DCACHE_LINE_SIZE
 - #define VTSS_OS_DCACHE_INVALIDATE(virt_addr, size) HAL_DCACHE_INVALIDATE(virt_addr, size)
 - #define VTSS_OS_DCACHE_FLUSH(virt_addr, size) HAL_DCACHE_STORE(virt_addr, size)
 - #define VTSS_OS_VIRT_TO_PHYS(addr) (u32)CYGARC_PHYSICAL_ADDRESS(addr)
 - #define VTSS_OS_BIG_ENDIAN
- VTSS_OS_BIG_ENDIAN: If undefined, we're running little endian. If defined we're running big endian.*
- #define VTSS_OS_ntohl(x) (x)
 - #define VTSS_OS_SCHEDULER_FLAGS cyg_uint32 __attribute__((unused))
 - #define VTSS_OS_SCHEDULER_LOCK(flags) cyg_scheduler_lock(__FILE__, __LINE__)
 - #define VTSS_OS_SCHEDULER_UNLOCK(flags) cyg_scheduler_unlock(__FILE__, __LINE__)
 - #define VTSS_OS_INTERRUPT_FLAGS NOT_NEEDED
 - #define VTSS_OS_INTERRUPT_DISABLE(flags) NOT_NEEDED
 - #define VTSS_OS_INTERRUPT_RESTORE(flags) NOT_NEEDED

Typedefs

- typedef cyg_tick_count_t vtss_mtimer_t

Functions

- long long int llabs (long long int val)
Obtain the absolute value of a long long integer.
- void * vtss_callout_malloc (size_t size, vtss_mem_flags_t flags)
Callout to allocate memory.
- void vtss_callout_free (void *ptr, vtss_mem_flags_t flags)
Callout to free memory.

8.25.1 Detailed Description

eCos OS API

This header file describes OS functions for eCos

8.25.2 Macro Definition Documentation

8.25.2.1 VTSS_MSLEEP

```
#define VTSS_MSLEEP(
    msec ) HAL_DELAY_US (msec*1000)
```

Sleep for "msec" milliseconds

Definition at line 50 of file vtss_os_ecos.h.

8.25.2.2 VTSS_NSLEEP

```
#define VTSS_NSLEEP(
    nsec ) HAL_DELAY_US( (nsec)/1000)
```

Sleep for "nsec" nanoseconds

Definition at line 51 of file vtss_os_ecos.h.

8.25.2.3 VTSS_MTIMER_START

```
#define VTSS_MTIMER_START(
    pTimer,
    msec ) *pTimer = cyg_current_time() + ((msec)/10) + 1
```

Starting Timer

Definition at line 54 of file vtss_os_ecos.h.

8.25.2.4 VTSS_MTIMER_TIMEOUT

```
#define VTSS_MTIMER_TIMEOUT(
    pTimer ) (cyg_current_time() > *(pTimer))
```

Timer timeout

Definition at line 55 of file vtss_os_ecos.h.

8.25.2.5 VTSS_MTIMER_CANCEL

```
#define VTSS_MTIMER_CANCEL(
    pTimer )
```

No action in this implementation.

Definition at line 56 of file vtss_os_ecos.h.

8.25.2.6 VTSS_TIME_OF_DAY

```
#define VTSS_TIME_OF_DAY(
    tod ) (tod.sec = (cyg_current_time() / CYGNUM_HAL_RTC_DENOMINATOR))
```

Time of day macro

Definition at line 63 of file vtss_os_ecos.h.

8.25.2.7 VTSS_DIV64

```
#define VTSS_DIV64(  
    dividend,  
    divisor ) ((dividend) / (divisor))
```

support for 64 bit division

Definition at line 73 of file vtss_os_ecos.h.

8.25.2.8 VTSS_MOD64

```
#define VTSS_MOD64(  
    dividend,  
    divisor ) ((dividend) % (divisor))
```

support for 64 bit division

Definition at line 74 of file vtss_os_ecos.h.

8.25.2.9 VTSS_LABS

```
#define VTSS_LABS(  
    arg ) labs(arg)
```

long to abs

Definition at line 75 of file vtss_os_ecos.h.

8.25.2.10 VTSS_LLabs

```
#define VTSS_LLabs(  
    arg ) llabs(arg)
```

long long to abs

Definition at line 76 of file vtss_os_ecos.h.

8.25.2.11 VTSS_OS_C TZ

```
#define VTSS_OS_C TZ(
    val32 ) ((val32) == 0 ? 32 : __builtin_ctz(val32))
```

Count trailing zeros of a 32-bit unsigned. Requirements/examples: `VTSS_OS_C TZ(0x00000001) = 0` `VTSS_OS_C TZ(0x80000000) = 31` `VTSS_OS_C TZ(0x00000000) >= 32` (if result is taken as unsigned; Most implementations return -1, and `(u32)(-1) >= 32`).

`__builtin_ctz()` is included in GCC 3.2.2 and later according to http://en.wikipedia.org/wiki/Find-first_set.

Note: `__builtin_ctz()` is undefined for zero input values.

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

8.25.2.12 VTSS_OS_C TZ64

```
#define VTSS_OS_C TZ64(
    val64 ) ((val64) == 0 ? 64 : __builtin_ctzll(val64))
```

Count trailing zeros of a 64-bit unsigned. Requirements/examples: `VTSS_OS_C TZ64(0x00000000_00000001) = 0` `VTSS_OS_C TZ64(0x00000000_80000000) = 31` `VTSS_OS_C TZ64(0x00000001_00000000) = 32` `VTSS_OS_C TZ64(0x80000000_00000000) = 63` `VTSS_OS_C TZ64(0x00000000_00000000) >= 64` (if result is taken as unsigned; Most implementations return -1, and `(u32)(-1) >= 64`).

Note: `__builtin_ctzll()` is undefined for zero input values.

Definition at line 102 of file `vtss_os_ecos.h`.

8.25.2.13 VTSS_OS_M ALLOC

```
#define VTSS_OS_M ALLOC(
    size,
    flags ) vtss_callout_malloc(size, flags)
```

Request OS to allocate `size` bytes of memory.

The first argument is the number of bytes that must be allocated. Type is `size_t`.

The second argument is a mask of flags that the implementation must obey. Type is `vtss_mem_flags_t`.

The returned pointer should be at least 8-byte aligned, to make it suitable for a struct.

Definition at line 135 of file `vtss_os_ecos.h`.

8.25.2.14 VTSS_OS_FREE

```
#define VTSS_OS_FREE(  
    ptr,  
    flags ) vtss_callout_free(ptr, flags)
```

Request OS to free memory previously allocated with [VTSS_OS_MALLOC\(\)](#).

The first argument is the pointer previously obtained with a call to [VTSS_OS_MALLOC\(\)](#). Type is void *.

The second argument is a mask of flags identical to those passed to [VTSS_OS_MALLOC\(\)](#) when the memory was requested.

Definition at line 149 of file vtss_os_ecos.h.

8.25.2.15 VTSS_OS RAND

```
#define VTSS_OS RAND( ) rand()
```

Wrap of call to rand() defined in stdlib.h

Definition at line 154 of file vtss_os_ecos.h.

8.25.2.16 VTSS_OS REORDER_BARRIER

```
#define VTSS_OS REORDER_BARRIER( ) HAL_REORDER_BARRIER()
```

The compiler may swap instructions to optimize performance of the final code (size- or speed-wise). When configuration of hardware is involved, it may not always be valid to swap two statements. Consider for instance the following two writes to two FDMA registers: Write the source address to the FDMA Enable the FDMA. To the compiler, these two writes can be executed in any order and still semantically yield the correct result, had it been normal RAM they were written to. But since they are written to actual hardware, it is crucial that they are executed in the correct order. The [VTSS_OS REORDER_BARRIER\(\)](#) macro should implement code that ensures that the compiler doesn't optimize across the barrier.

Definition at line 173 of file vtss_os_ecos.h.

8.25.2.17 VTSS_OS_COMPILER_ATTRIBUTE_ALIGNED

```
#define VTSS_OS_COMPILER_ATTRIBUTE_ALIGNED(  
    x ) __attribute__ ((aligned(x)))
```

In some special cases, it is of utmost importance that a certain variable has a certain memory alignment. Applications for this is e.g. placing variables on cache-line boundaries.

Definition at line 180 of file vtss_os_ecos.h.

8.25.2.18 VTSS_OS_DCACHE_LINE_SIZE_BYTES

```
#define VTSS_OS_DCACHE_LINE_SIZE_BYTES HAL_DCACHE_LINE_SIZE
```

The number of bytes one DCache-line is made up of.

Definition at line 189 of file vtss_os_ecos.h.

8.25.2.19 VTSS_OS_DCACHE_INVALIDATE

```
#define VTSS_OS_DCACHE_INVALIDATE(  
    virt_addr,  
    size ) HAL_DCACHE_INVALIDATE(virt_addr, size)
```

Invalidate @size bytes at virtual address @virt_addr of the DCache. After invalidation, the invalidated area will be fetched from RAM.

Definition at line 195 of file vtss_os_ecos.h.

8.25.2.20 VTSS_OS_DCACHE_FLUSH

```
#define VTSS_OS_DCACHE_FLUSH(  
    virt_addr,  
    size ) HAL_DCACHE_STORE(virt_addr, size)
```

Force a write of @size bytes of dirty cache lines to RAM starting at virtual address @virt_addr.

Definition at line 201 of file vtss_os_ecos.h.

8.25.2.21 VTSS_OS_VIRT_TO_PHYS

```
#define VTSS_OS_VIRT_TO_PHYS(  
    addr ) (u32)CYGARC_PHYSICAL_ADDRESS(addr)
```

Macro that implements the conversion from a virtual to a physical address. In OSs with a flat memory layout, this could be as simple as (u32)(addr).

Definition at line 208 of file vtss_os_ecos.h.

8.25.2.22 VTSS_OS_BIG_ENDIAN

```
#define VTSS_OS_BIG_ENDIAN
```

VTSS_OS_BIG_ENDIAN: If undefined, we're running little endian. If defined we're running big endian.

We're running big endian

Definition at line 221 of file vtss_os_ecos.h.

8.25.2.23 VTSS_OS_NTOHL

```
#define VTSS_OS_NTOHL( x ) (x)
```

Convert from network to host order

Definition at line 222 of file vtss_os_ecos.h.

8.25.2.24 VTSS_OS_SCHEDULER_FLAGS

```
#define VTSS_OS_SCHEDULER_FLAGS cyg_uint32 __attribute__((unused))
```

VTSS_OS_SCHEDULER_FLAGS VTSS_OS_SCHEDULER_LOCK(flags) VTSS_OS_SCHEDULER_UNLOCK(flags)

These functions are called by API code that consists of a user-level part and a deferred interrupt handler part. Only the user-level part will call the `VTSS_OS_SCHEDULER_LOCK()`/`UNLOCK()` functions, since it is assumed that the deferred interrupt handler will have atomic access throughout its execution. Each module within the API that contains such functionality will have an option to call either the `VTSS_OS_SCHEDULER_(UN)LOCK()` functions or the `VTSS_OS_INTERRUPT_DISABLE()`/`RESTORE()` functions. The `attribute((unused))` ensures that we don't get compiler warnings.

Definition at line 248 of file vtss_os_ecos.h.

8.25.2.25 VTSS_OS_SCHEDULER_LOCK

```
#define VTSS_OS_SCHEDULER_LOCK( flags ) cyg_scheduler_lock(__FILE__, __LINE__)
```

Lock scheduler.

Definition at line 252 of file vtss_os_ecos.h.

8.25.2.26 VTSS_OS_SCHEDULER_UNLOCK

```
#define VTSS_OS_SCHEDULER_UNLOCK(  
    flags ) cyg_scheduler_unlock(__FILE__, __LINE__)
```

Unlock scheduler.

Definition at line 256 of file vtss_os_ecos.h.

8.25.2.27 VTSS_OS_INTERRUPT_FLAGS

```
#define VTSS_OS_INTERRUPT_FLAGS NOT_NEEDED
```

VTSS_OS_INTERRUPT_FLAGS [VTSS_OS_INTERRUPT_DISABLE\(flags\)](#) [VTSS_OS_INTERRUPT_RESTORE\(flags\)](#)
These functions are called by API code that consists of a user-level part and an interrupt handler part executing directly in interrupt context. Only the user-level part will call the [VTSS_OS_INTERRUPT_DISABLE\(\)](#)/[RESTORE\(\)](#) functions, since it is assumed that the interrupt handler will have atomic access throughout its execution. Each module within the API that contains such functionality will have an option to call either the [VTSS_OS_SCHEDULER_\(UN\)LOCK\(\)](#) functions or the [VTSS_OS_INTERRUPT_DISABLE\(\)](#)/[RESTORE\(\)](#) functions. Not needed in eCos, since all interrupt handlers will be called in deferred context.

Definition at line 272 of file vtss_os_ecos.h.

8.25.2.28 VTSS_OS_INTERRUPT_DISABLE

```
#define VTSS_OS_INTERRUPT_DISABLE(  
    flags ) NOT_NEEDED
```

Disable interrupts.

Definition at line 276 of file vtss_os_ecos.h.

8.25.2.29 VTSS_OS_INTERRUPT_RESTORE

```
#define VTSS_OS_INTERRUPT_RESTORE(  
    flags ) NOT_NEEDED
```

Enable interrupts.

Definition at line 280 of file vtss_os_ecos.h.

8.25.3 Typedef Documentation

8.25.3.1 vtss_mtimer_t

```
typedef cyg_tick_count_t vtss_mtimer_t
```

Timer

Definition at line 53 of file vtss_os_ecos.h.

8.25.4 Function Documentation

8.25.4.1 llabs()

```
long long int llabs (
    long long int val )
```

Obtain the absolute value of a long long integer.

Parameters

<i>val</i>	[IN] The value to convert to absolute value.
------------	----------------------------------------------

Returns

The absolute value of val.

8.25.4.2 vtss_callout_malloc()

```
void* vtss_callout_malloc (
    size_t size,
    vtss_mem_flags_t flags )
```

Callout to allocate memory.

[IN]/[OUT] seen from called function.

Parameters

<i>size</i>	[IN] Number of bytes to allocate.
<i>flags</i>	[IN] See vtss_mem_flags_t for details.

Returns

Pointer to allocated area.

8.25.4.3 `vtss_callout_free()`

```
void vtss_callout_free (
    void * ptr,
    vtss_mem_flags_t flags )
```

Callout to free memory.

[IN]/[OUT] seen from called function.

Parameters

<code>ptr</code>	[IN] Pointer previously obtained with call to vtss_callout_malloc() .
<code>flags</code>	[IN] See <code>vtss_mem_flags_t</code> for details.

8.26 `vtss_api/include/vtss_os_linux.h` File Reference

Linux OS API.

```
#include <endian.h>
#include <asm/byteorder.h>
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>
#include <errno.h>
#include <time.h>
#include <sys/time.h>
```

Data Structures

- struct `vtss_mtimer_t`
Timer structure.
- struct `vtss_timeofday_t`
Time of day structure.

Macros

- `#define VTSS_OS_BIG_ENDIAN`
VTSS_OS_BIG_ENDIAN: If undefined, we're running little endian. If defined we're running big endian.
- `#define VTSS_OS_NTOHL(x) __be32_to_cpu(x)`
- `#define VTSS_NSLEEP(nsec)`
- `#define VTSS_MSLEEP(msec)`
- `#define VTSS_MTIMER_START(timer, msec)`
- `#define VTSS_MTIMER_TIMEOUT(timer) (gettimeofday(&((timer)->now),NULL)==0 && timercmp(&((timer)->now),&((timer)->timeout),>))`
- `#define VTSS_MTIMER_CANCEL(timer)`
- `#define VTSS_TIME_OF_DAY(tod)`

- #define `VTSS_OS_SCHEDULER_FLAGS` int
- #define `VTSS_OS_SCHEDULER_LOCK`(flags) do {flags = flags;} while (0);
- #define `VTSS_OS_SCHEDULER_UNLOCK`(flags) do {flags = flags;} while (0);
- #define `VTSS_DIV64`(dividend, divisor) ((dividend) / (divisor))
- #define `VTSS_MOD64`(dividend, divisor) ((dividend) % (divisor))
- #define `VTSS_LABS`(arg) labs(arg)
- #define `VTSS_LLabs`(arg) llabs(arg)
- #define `VTSS_OS_Ctz`(val32) ((val32) == 0 ? 32 : __builtin_ctzl((unsigned long)val32))
- #define `VTSS_OS_Ctz64`(val64)
- #define `VTSS_OS_MALLOC`(size, flags) malloc(size)
- #define `VTSS_OS_FREE`(ptr, flags) free(ptr)
- #define `VTSS_OS_RAND`() rand()

8.26.1 Detailed Description

Linux OS API.

This header file describes OS functions for Linux

8.26.2 Macro Definition Documentation

8.26.2.1 VTSS_OS_BIG_ENDIAN

```
#define VTSS_OS_BIG_ENDIAN
```

`VTSS_OS_BIG_ENDIAN`: If undefined, we're running little endian. If defined we're running big endian.

We're running big endian

Definition at line 46 of file `vtss_os_linux.h`.

8.26.2.2 VTSS_OS_NTOHL

```
#define VTSS_OS_NTOHL( x ) __be32_to_cpu(x)
```

Convert a 32-bit value from network to host order

Definition at line 49 of file `vtss_os_linux.h`.

8.26.2.3 VTSS_NSLEEP

```
#define VTSS_NSLEEP( nsec )
```

Value:

```
{
    struct timespec ts;
    ts.tv_sec = 0;
    ts.tv_nsec = nsec;
    while(nanosleep(&ts, &ts) == -1 && errno == EINTR) {
    }
}
```

Sleep for

Parameters

<i>nsec</i>	nanoseconds
-------------	-------------

Definition at line 69 of file vtss_os_linux.h.

8.26.2.4 VTSS_MSLEEP

```
#define VTSS_MSLEEP(
    msec )
```

Value:

```
{
    struct timespec ts;                                \
    ts.tv_sec = msec / 1000;                          \
    ts.tv_nsec = (msec % 1000) * 1000000;             \
    while(nanosleep(&ts, &ts) == -1 && errno == EINTR) { \
        }                                              \
}
```

Sleep for

Parameters

<i>msec</i>	milliseconds
-------------	--------------

Definition at line 78 of file vtss_os_linux.h.

8.26.2.5 VTSS_MTIMER_START

```
#define VTSS_MTIMER_START(
    timer,
    msec )
```

Value:

```
{
    (void) gettimeofday(&((timer)->timeout),NULL);      \
    (timer)->timeout.tv_usec+=msec*1000; \
    if ((timer)->timeout.tv_usec>=1000000) { (timer)->timeout.tv_sec+=(timer)->timeout.tv_usec/1000000; \
        (timer)->timeout.tv_usec%=1000000; } \
}
```

Start timer

Definition at line 93 of file vtss_os_linux.h.

8.26.2.6 VTSS_MTIMER_TIMEOUT

```
#define VTSS_MTIMER_TIMEOUT(
    timer ) (gettimeofday(&((timer)->now),NULL)==0 && timercmp(&((timer)->now),&((timer)->timeout),
```

Timer timeout

Definition at line 103 of file vtss_os_linux.h.

8.26.2.7 VTSS_MTIMER_CANCEL

```
#define VTSS_MTIMER_CANCEL(
    timer )
```

No action in this implementation. Time of day struct

Definition at line 105 of file vtss_os_linux.h.

8.26.2.8 VTSS_TIME_OF_DAY

```
#define VTSS_TIME_OF_DAY(
    tod )
```

Value:

```
{ \
    struct timeval tve; \
    (void) gettimeofday(&tve,NULL); \
    tod.sec = tve.tv_sec; \
}
```

Time of day macro

Definition at line 112 of file vtss_os_linux.h.

8.26.2.9 VTSS_OS_SCHEDULER_FLAGS

```
#define VTSS_OS_SCHEDULER_FLAGS int
```

VTSS_OS_SCHEDULER_FLAGS VTSS_OS_SCHEDULER_LOCK(flags) VTSS_OS_SCHEDULER_UNLOCK(flags)
These functions are called by API code that consists of a user-level part and a deferred interrupt handler part. Only the user-level part will call the VTSS_OS_SCHEDULER_LOCK() /UNLOCK() functions, since it is assumed that the deferred interrupt handler will have atomic access throughout its execution. Each module within the API that contains such functionality will have an option to call either the VTSS_OS_SCHEDULER_(UN)LOCK() functions or the VTSS_OS_INTERRUPT_DISABLE() /RESTORE() functions.

Definition at line 138 of file vtss_os_linux.h.

8.26.2.10 VTSS_OS_SCHEDULER_LOCK

```
#define VTSS_OS_SCHEDULER_LOCK(  
    flags ) do {flags = flags;} while (0);
```

Lock scheduler

Definition at line 139 of file vtss_os_linux.h.

8.26.2.11 VTSS_OS_SCHEDULER_UNLOCK

```
#define VTSS_OS_SCHEDULER_UNLOCK(  
    flags ) do {flags = flags;} while (0);
```

Unlock scheduler

Definition at line 140 of file vtss_os_linux.h.

8.26.2.12 VTSS_DIV64

```
#define VTSS_DIV64(  
    dividend,  
    divisor ) ((dividend) / (divisor))
```

VTSS_DIV64 - perform 64/32 bit division yielding 32 bit (at least) output

Definition at line 145 of file vtss_os_linux.h.

8.26.2.13 VTSS_MOD64

```
#define VTSS_MOD64(  
    dividend,  
    divisor ) ((dividend) % (divisor))
```

VTSS_MOD64 - perform 64/32 bit modulus yielding 32 bit (at least) output

Definition at line 149 of file vtss_os_linux.h.

8.26.2.14 VTSS_LABS

```
#define VTSS_LABS(
    arg ) labs(arg)
```

VTSS_LABS - perform abs() on long

Definition at line 153 of file vtss_os_linux.h.

8.26.2.15 VTSS_LLABS

```
#define VTSS_LLABS(
    arg ) llabs(arg)
```

VTSS_LLABS - perform abs() on long long

Definition at line 158 of file vtss_os_linux.h.

8.26.2.16 VTSS_OS_CTZ

```
#define VTSS_OS_CTZ(
    val32 ) ((val32) == 0 ? 32 : __builtin_ctzl((unsigned long)val32))
```

VTSS_OS_CTZ(val32)

Count trailing zeros of a 32-bit unsigned. Requirements/examples: `VTSS_OS_CTZ(0x00000001) = 0` `VTSS_OS_CTZ(0x80000000) = 31` `VTSS_OS_CTZ(0x00000000) >= 32` (if result is taken as unsigned; Most implementations return -1, and `(u32)(-1) >= 32`).

Parameters

<code>val32</code>	The value to decode
--------------------	---------------------

Returns

Number of trailing zeroes - or - the bit index of the lowest bit set in the input given.

Note

`__builtin_ctz()` is included in GCC 3.2.2 and later according to http://en.wikipedia.org/wiki/Find_first_set.

Note: `__builtin_ctzl()` is undefined for zero input values.

Definition at line 370 of file vtss_os_linux.h.

8.26.2.17 VTSS_OS_C TZ64

```
#define VTSS_OS_C TZ64( \
    val64 )
```

Value:

```
(({ \
    u32 _r = VTSS_OS_C TZ((u32)(val64)); \
    (val64) == 0 ? 64 : \
    _r < 32 ? _r : 32 + VTSS_OS_C TZ((u32)((val64) >> 32)); \
}))
```

Count trailing zeros of a 64-bit unsigned. Requirements/examples: [VTSS_OS_C TZ64\(0x00000000_00000001\)](#) = 0 [VTSS_OS_C TZ64\(0x00000000_80000000\)](#) = 31 [VTSS_OS_C TZ64\(0x00000001_00000000\)](#) = 32 [VTSS_OS_C TZ64\(0x80000000_00000000\)](#) = 63 [VTSS_OS_C TZ64\(0x00000000_00000000\)](#) >= 64 (if result is taken as unsigned; Most implementations return -1, and (u32)(-1) >= 64).

Definition at line 381 of file vtss_os_linux.h.

8.26.2.18 VTSS_OS_M ALLOC

```
#define VTSS_OS_M ALLOC( \
    size, \
    flags ) malloc(size)
```

Request OS to allocate size bytes of memory.

The first argument is the number of bytes that must be allocated. Type is size_t.

The second argument is a mask of flags that the implementation must obey. Type is vtss_mem_flags_t.

The returned pointer should be at least 8-byte aligned, to make it suitable for a struct.

Definition at line 425 of file vtss_os_linux.h.

8.26.2.19 VTSS_OS_F REE

```
#define VTSS_OS_F REE( \
    ptr, \
    flags ) free(ptr)
```

Request OS to free memory previously allocated with [VTSS_OS_M ALLOC\(\)](#).

The first argument is the pointer previously obtained with a call to [VTSS_OS_M ALLOC\(\)](#). Type is void *.

The second argument is a mask of flags identical to those passed to [VTSS_OS_M ALLOC\(\)](#) when the memory was requested.

Definition at line 443 of file vtss_os_linux.h.

8.26.2.20 VTSS_OS_RAND

```
#define VTSS_OS_RAND( ) rand()
```

Wrap of call to rand() defined in stdlib.h

Definition at line 449 of file vtss_os_linux.h.

8.27 vtss_api/include/vtss_otn_api.h File Reference

OTN API.

```
#include <vtss/api/types.h>
```

8.27.1 Detailed Description

OTN API.

8.28 vtss_api/include/vtss_packet_api.h File Reference

Packet API.

```
#include <vtss/api/types.h>
#include <vtss_l2_api.h>
```

Data Structures

- struct [vtss_npi_conf_t](#)
NPI configuration.
- struct [vtss_packet_rx_queue_npi_conf_t](#)
CPU Rx queue NPI configuration.
- struct [vtss_packet_rx_queue_conf_t](#)
CPU Rx queue configuration.
- struct [vtss_packet_rx_reg_t](#)
CPU Rx packet registration.
- struct [vtss_packet_rx_queue_map_t](#)
CPU Rx queue map.
- struct [vtss_packet_rx_conf_t](#)
CPU Rx configuration.
- struct [vtss_vstax_rx_header_t](#)
VStaX frame header used for reception.
- struct [vtss_tci_t](#)
Tag Control Information (according to IEEE 802.1Q)
- struct [vtss_packet_rx_header_t](#)
System frame header describing received frame.

- struct [vtss_packet_frame_info_t](#)
Information about frame.
- struct [vtss_packet_port_info_t](#)
Port info structure.
- struct [vtss_packet_port_filter_t](#)
Packet information for each port.
- struct [vtss_vstax_tx_header_t](#)
VStaX frame header used for transmission.
- struct [vtss_packet_rx_meta_t](#)
Input structure to [vtss_packet_rx_hdr_decode\(\)](#).
- struct [vtss_packet_rx_info_t](#)
Decoded extraction header properties.
- struct [vtss_packet_tx_info_t](#)
Injection Properties.
- struct [vtss_packet_tx_ifh_t](#)
Compiled Tx Frame Header.
- struct [vtss_packet_dma_conf_t](#)

Macros

- #define VTSS_PRIO_SUPER VTSS_PRIO_END
- #define VTSS_VSTAX_TTL_PORT 0
- #define VTSS_VSTAX_HDR_SIZE 12
- #define VTSS_JR1_PACKET_HDR_SIZE_BYTES 52
- #define VTSS_JR2_PACKET_HDR_SIZE_BYTES 32
- #define VTSS_SVL_PACKET_HDR_SIZE_BYTES 20
- #define VTSS_L26_PACKET_HDR_SIZE_BYTES 16
- #define VTSS_PACKET_HDR_SIZE_BYTES VTSS_JR1_PACKET_HDR_SIZE_BYTES
- #define VTSS_SVL_RX_IFH_SIZE 16
- #define VTSS_JR1_RX_IFH_SIZE 24
- #define VTSS_L26_RX_IFH_SIZE 8
- #define VTSS_JR2_RX_IFH_SIZE 28
- #define VTSS_PACKET_TX_IFH_MAX 24

Typedefs

- typedef u32 [vtss_packet_rx_queue_size_t](#)
CPU Rx queue buffer size in bytes.
- typedef u32 [vtss_vstax_ttl_t](#)
VStaX TTL value, 0-31.

Enumerations

- enum `vtss_packet_filter_t` { `VTSS_PACKET_FILTER_DISCARD`, `VTSS_PACKET_FILTER_TAGGED`, `VTSS_PACKET_FILTER_UNTAGGED` }

CPU filter.
- enum `vtss_vstax_fwd_mode_t` { `VTSS_VSTAX_FWD_MODE_LOOKUP` = 0, `VTSS_VSTAX_FWD_MODE_UPSID_PORT` = 2, `VTSS_VSTAX_FWD_MODE_CPU_UPSID` = 5, `VTSS_VSTAX_FWD_MODE_CPU_ALL` = 6 }

VStaX frame forward mode.
- enum `vtss_packet_oam_type_t` {
 `VTSS_PACKET_OAM_TYPE_NONE` = 0, `VTSS_PACKET_OAM_TYPE_CCM`, `VTSS_PACKET_OAM_TYPE_CCM_LM`, `VTSS_PACKET_OAM_TYPE_LBM`,
`VTSS_PACKET_OAM_TYPE_LBR`, `VTSS_PACKET_OAM_TYPE_LMM`, `VTSS_PACKET_OAM_TYPE_LMR`,
`VTSS_PACKET_OAM_TYPE_DMM`,
`VTSS_PACKET_OAM_TYPE_DMR`, `VTSS_PACKET_OAM_TYPE_1DM`, `VTSS_PACKET_OAM_TYPE_LTM`,
`VTSS_PACKET_OAM_TYPE_LTR`,
`VTSS_PACKET_OAM_TYPE_GENERIC` }
- enum `vtss_packet_ptp_action_t` {
 `VTSS_PACKET_PTP_ACTION_NONE` = 0, `VTSS_PACKET_PTP_ACTION_ONE_STEP`, `VTSS_PACKET_PTP_ACTION_TWO_STEP`,
`VTSS_PACKET_PTP_ACTION_ONE_AND_TWO_STEP`,
`VTSS_PACKET_PTP_ACTION_ORIGIN_TIMESTAMP` }
- enum `vtss_tag_type_t` { `VTSS_TAG_TYPE_UNTAGGED` = 0, `VTSS_TAG_TYPE_C_TAGGED`, `VTSS_TAG_TYPE_S_TAGGED`,
`VTSS_TAG_TYPE_S_CUSTOM_TAGGED` }
- enum `vtss_packet_rx_hints_t` { `VTSS_PACKET_RX_HINTS_NONE` = 0x00, `VTSS_PACKET_RX_HINTS_VLAN_TAG_MISMATCH` = 0x01, `VTSS_PACKET_RX_HINTS_VLAN_FRAME_MISMATCH` = 0x02, `VTSS_PACKET_RX_HINTS_VID_MISMATCH` = 0x04 }

Provides additional info on decoded extraction header.
- enum `vtss_packet_tx_vstax_t` { `VTSS_PACKET_TX_VSTAX_NONE` = 0, `VTSS_PACKET_TX_VSTAX_BIN`, `VTSS_PACKET_TX_VSTAX_SYM` }

Transmit frames with VStaX header, and if so, how is it specified?

Functions

- `vtss_rc vtss_npi_conf_get` (const `vtss_inst_t` inst, `vtss_npi_conf_t` *const conf)

Get NPI configuration.
- `vtss_rc vtss_npi_conf_set` (const `vtss_inst_t` inst, const `vtss_npi_conf_t` *const conf)

Set NPI configuration.
- `vtss_rc vtss_packet_rx_conf_get` (const `vtss_inst_t` inst, `vtss_packet_rx_conf_t` *const conf)

Get Packet Rx configuration.
- `vtss_rc vtss_packet_rx_conf_set` (const `vtss_inst_t` inst, const `vtss_packet_rx_conf_t` *const conf)

Set CPU Rx queue configuration.
- `vtss_rc vtss_packet_rx_port_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_packet_rx_port_conf_t` *const conf)

Get packet configuration for port.
- `vtss_rc vtss_packet_rx_port_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_packet_rx_port_conf_t` *const conf)

Set packet configuration for port.
- `vtss_rc vtss_packet_rx_frame_get` (const `vtss_inst_t` inst, const `vtss_packet_rx_queue_t` queue_no, `vtss_packet_rx_header_t` *const header, `u8` *const frame, const `u32` length)

Copy a received frame from a CPU queue.
- `vtss_rc vtss_packet_rx_frame_get_raw` (const `vtss_inst_t` inst, `u8` *const data, const `u32` buflen, `u32` *const ifhlen, `u32` *const frrlen)

Copy a received frame from a CPU queue - with IFH.
- `vtss_rc vtss_packet_rx_frame_discard` (const `vtss_inst_t` inst, const `vtss_packet_rx_queue_t` queue_no)

- `vtss_rc vtss_packet_tx_frame_port` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `u8` *const frame, const `u32` length)

Send frame unmodified on port.
- `vtss_rc vtss_packet_tx_frame_port_vlan` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_vid_t` vid, const `u8` *const frame, const `u32` length)

Send frame on port using egress rules.
- `vtss_rc vtss_packet_tx_frame_vlan` (const `vtss_inst_t` inst, const `vtss_vid_t` vid, const `u8` *const frame, const `u32` length)

Send frame on VLAN using egress rules.
- void `vtss_packet_frame_info_init` (`vtss_packet_frame_info_t` *const info)

Initialize filter information to default values.
- `vtss_rc vtss_packet_frame_filter` (const `vtss_inst_t` inst, const `vtss_packet_frame_info_t` *const info, `vtss_packet_filter_t` *const filter)

Determine ingress/egress filter for frame. The function may be used for ingress filtering, egress filtering or both.
- `vtss_rc vtss_packet_port_info_init` (`vtss_packet_port_info_t` *const info)

Initialize filter information to default values.
- `vtss_rc vtss_packet_port_filter_get` (const `vtss_inst_t` inst, const `vtss_packet_port_info_t` *const info, `vtss_packet_port_filter_t` filter[`VTSS_PORT_ARRAY_SIZE`])

Determine ingress/egress filter for frame per port. The function may be used for ingress filtering, egress filtering or both.
- `vtss_rc vtss_packet_tx_frame_vstax` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_vstax_tx_header_t` *const header, const `u8` *const frame, const `u32` length)

Send frame on VStaX port.
- `vtss_rc vtss_packet_vstax_header2frame` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_vstax_tx_header_t` *const header, `u8` *const frame)

Build 12 bytes VStaX frame header.
- `vtss_rc vtss_packet_vstax_frame2header` (const `vtss_inst_t` inst, const `u8` *const frame, `vtss_vstax_rx_header_t` *const header)

Extract 12 bytes VStaX header.
- `vtss_rc vtss_packet_rx_hdr_decode` (const `vtss_inst_t` inst, const `vtss_packet_rx_meta_t` *const meta, const `u8` hdr[`VTSS_PACKET_HDR_SIZE_BYTES`], `vtss_packet_rx_info_t` *const info)

Decode binary extraction/Rx header.
- `vtss_rc vtss_packet_tx_hdr_encode` (const `vtss_inst_t` inst, const `vtss_packet_tx_info_t` *const info, `u8` *const bin_hdr, `u32` *const bin_hdr_len)

Compose binary injection/Tx header.
- `vtss_rc vtss_packet_tx_hdr_compile` (const `vtss_inst_t` inst, const `vtss_packet_tx_info_t` *const info, `vtss_packet_tx_ifh_t` *const ifh)

Compile Tx Frame Header.
- `vtss_rc vtss_packet_tx_frame` (const `vtss_inst_t` inst, const `vtss_packet_tx_ifh_t` *const ifh, const `u8` *const frame, const `u32` length)

Send frame unmodified on port with pre-compiled IFH.
- `vtss_rc vtss_packet_tx_info_init` (const `vtss_inst_t` inst, `vtss_packet_tx_info_t` *const info)

Initialize a Tx info structure.
- `vtss_rc vtss_packet_dma_conf_get` (const `vtss_inst_t` inst, `vtss_packet_dma_conf_t` *const conf)

Retreive packet DMA configuration.
- `vtss_rc vtss_packet_dma_conf_set` (const `vtss_inst_t` inst, const `vtss_packet_dma_conf_t` *const conf)

Set packet DMA configuration.
- `vtss_rc vtss_packet_dma_offset` (const `vtss_inst_t` inst, `BOOL` extraction, `u32` *offset)

Retreive the register offset for extraction/injection DMA.

8.28.1 Detailed Description

Packet API.

This header file describes CPU Rx/Tx packet functions.

8.28.2 Macro Definition Documentation

8.28.2.1 VTSS_PRIO_SUPER

```
#define VTSS_PRIO_SUPER VTSS_PRIO_END
```

Super priority

Definition at line 429 of file vtss_packet_api.h.

8.28.2.2 VTSS_VSTAX_TTL_PORT

```
#define VTSS_VSTAX_TTL_PORT 0
```

TTL value configured for port is used

Definition at line 448 of file vtss_packet_api.h.

8.28.2.3 VTSS_VSTAX_HDR_SIZE

```
#define VTSS_VSTAX_HDR_SIZE 12
```

VStaX header size

Definition at line 490 of file vtss_packet_api.h.

8.28.2.4 VTSS_JR1_PACKET_HDR_SIZE_BYTES

```
#define VTSS_JR1_PACKET_HDR_SIZE_BYTES 52
```

Max header size. Worst case: XTR = INJ (XTR: 24 bytes for IFH + 4 bytes for queue number + 24 bytes for Rx super-prio artificial IFH, INJ: 24 bytes for IFH + 4 bytes VLAN tag + 24 bytes for signature IFH for multicast injections)

Definition at line 694 of file vtss_packet_api.h.

8.28.2.5 VTSS_JR2_PACKET_HDR_SIZE_BYTES

```
#define VTSS_JR2_PACKET_HDR_SIZE_BYTES 32
```

Max header size. Worst case: INJ (28 bytes for IFH + 4 bytes for VLAN tag)

Definition at line 695 of file vtss_packet_api.h.

8.28.2.6 VTSS_SVL_PACKET_HDR_SIZE_BYTES

```
#define VTSS_SVL_PACKET_HDR_SIZE_BYTES 20
```

Max header size. Worst case: INJ (16 bytes for IFH + 4 bytes for VLAN tag)

Definition at line 696 of file vtss_packet_api.h.

8.28.2.7 VTSS_L26_PACKET_HDR_SIZE_BYTES

```
#define VTSS_L26_PACKET_HDR_SIZE_BYTES 16
```

Max header size. Worst case: INJ (8 bytes for IFH + 4 for timestamp + 4 for VLAN tag)

Definition at line 697 of file vtss_packet_api.h.

8.28.2.8 VTSS_PACKET_HDR_SIZE_BYTES

```
#define VTSS_PACKET_HDR_SIZE_BYTES VTSS_JR1_PACKET_HDR_SIZE_BYTES
```

Maximum header size. This define is only useful if you only compile for one target.

Definition at line 701 of file vtss_packet_api.h.

8.28.2.9 VTSS_SVL_RX_IFH_SIZE

```
#define VTSS_SVL_RX_IFH_SIZE 16
```

Serval1 Rx IFH size is 16 bytes

Definition at line 710 of file vtss_packet_api.h.

8.28.2.10 VTSS_JR1_RX_IFH_SIZE

```
#define VTSS_JR1_RX_IFH_SIZE 24
```

Jaguar1 Rx IFH size is 24 bytes

Definition at line 711 of file vtss_packet_api.h.

8.28.2.11 VTSS_L26_RX_IFH_SIZE

```
#define VTSS_L26_RX_IFH_SIZE 8
```

Luton26 Rx IFH size is 8 bytes

Definition at line 712 of file vtss_packet_api.h.

8.28.2.12 VTSS_JR2_RX_IFH_SIZE

```
#define VTSS_JR2_RX_IFH_SIZE 28
```

Jaguar2 Rx IFH size is 28 bytes

Definition at line 713 of file vtss_packet_api.h.

8.28.2.13 VTSS_PACKET_TX_IFH_MAX

```
#define VTSS_PACKET_TX_IFH_MAX 24
```

Tx IFH byte length (Constant)

Definition at line 2056 of file vtss_packet_api.h.

8.28.3 Enumeration Type Documentation

8.28.3.1 vtss_packet_filter_t

```
enum vtss_packet_filter_t
```

CPU filter.

Enumerator

VTSS_PACKET_FILTER_DISCARD	Discard
VTSS_PACKET_FILTER_TAGGED	Tagged transmission
VTSS_PACKET_FILTER_UNTAGGED	Untagged transmission

Definition at line 368 of file vtss_packet_api.h.

8.28.3.2 vtss_vstax_fwd_mode_t

```
enum vtss_vstax_fwd_mode_t
```

VStaX frame forward mode.

Enumerator

VTSS_VSTAX_FWD_MODE_LOOKUP	Local lookup in all units
VTSS_VSTAX_FWD_MODE_UPSID_PORT	Physical port on UPSID
VTSS_VSTAX_FWD_MODE_CPU_UPSID	CPU queue on UPSID
VTSS_VSTAX_FWD_MODE_CPU_ALL	CPU queue on all UPSIDs

Definition at line 435 of file vtss_packet_api.h.

8.28.3.3 vtss_packet_oam_type_t

```
enum vtss_packet_oam_type_t
```

OAM types to be used when encoding an injection header.

Enumerator

VTSS_PACKET_OAM_TYPE_NONE	No-op
VTSS_PACKET_OAM_TYPE_CCM	Continuity Check Message
VTSS_PACKET_OAM_TYPE_CCM_LM	Continuity Check Message with Loss Measurement information
VTSS_PACKET_OAM_TYPE_LBM	Loopback Message
VTSS_PACKET_OAM_TYPE_LBR	Loopback Reply
VTSS_PACKET_OAM_TYPE_LMM	Loss Measurement Message
VTSS_PACKET_OAM_TYPE_LMR	Loss Measurement Reply
VTSS_PACKET_OAM_TYPE_DMM	Delay Measurement Message
VTSS_PACKET_OAM_TYPE_DMR	Delay Measurement Reply
VTSS_PACKET_OAM_TYPE_1DM	A.k.a. SDM, One-Way Delay Measurement
VTSS_PACKET_OAM_TYPE_LTM	Link Trace message
VTSS_PACKET_OAM_TYPE_LTR	Link Trace Reply
VTSS_PACKET_OAM_TYPE_GENERIC	Generic OAM type

Definition at line 524 of file vtss_packet_api.h.

8.28.3.4 vtss_packet_ptp_action_t

enum [vtss_packet_ptp_action_t](#)

PTP actions used when encoding an injection header.

Enumerator

VTSS_PACKET_PTP_ACTION_NONE	No-op
VTSS_PACKET_PTP_ACTION_ONE_STEP	One-step PTP
VTSS_PACKET_PTP_ACTION_TWO_STEP	Two-step PTP
VTSS_PACKET_PTP_ACTION_ONE_AND_TWO_STEP	Both one- and two-step PTP
VTSS_PACKET_PTP_ACTION_ORIGIN_TIMESTAMPAMP	Update time-of-day in PTP frame's originTimestamp field

Definition at line 543 of file vtss_packet_api.h.

8.28.3.5 vtss_tag_type_t

enum [vtss_tag_type_t](#)

Tag type the frame was received with.

Enumerator

VTSS_TAG_TYPE_UNTAGGED	Frame was received untagged or on an unaware port or with a tag that didn't match the port type.
VTSS_TAG_TYPE_C_TAGGED	Frame was received with a C-tag
VTSS_TAG_TYPE_S_TAGGED	Frame was received with an S-tag
VTSS_TAG_TYPE_S_CUSTOM_TAGGED	Frame was received with a custom S-tag

Definition at line 554 of file vtss_packet_api.h.

8.28.3.6 vtss_packet_rx_hints_t

enum [vtss_packet_rx_hints_t](#)

Provides additional info on decoded extraction header.

In some cases, a frame received by the CPU should be dropped. This cannot always be detected by just looking at the decoded extraction header.

This enum provides the caller of [vtss_packet_rx_hdr_decode\(\)](#) with a means to determine whether to drop or forward the frame further to the application. Whether to actually drop it is fully up to the application.

The individual enums may be combined into a mask.

Enumerator

VTSS_PACKET_RX_HINTS_NONE	No hints.
VTSS_PACKET_RX_HINTS_VLAN_TAG_MISMATCH TCH	<p>If a frame is received on a C-port with a "foreign" tag (i.e. an S-tag or S-custom-tag), the frame should possibly be dropped. This is indicated with this enum being member of the hints flags.</p> <p>The same goes for frames received on S-ports or S-custom-ports with "foreign" tags.</p> <p>Can be set by:</p> <p>Luton26: Y Jaguar1: Y Serval: Y Jaguar2: Y Serval2: Y ServalT: Y</p>
VTSS_PACKET_RX_HINTS_VLAN_FRAME_MISMATCH	<p>If a tagged frame is received on a port that only should accept untagged frames or if an untagged frame is received on a port that should only accept tagged frames, then this will be set. In general, the application should not drop such frames, because e.g. BPDUs are normally untagged.</p> <p>Can be set by:</p> <p>Luton26: Y Jaguar1: Y Serval: Y Jaguar2: Y Serval2: Y ServalT: Y</p>
VTSS_PACKET_RX_HINTS_VID_MISMATCH	<p>If a frame gets classified to a VLAN ID on a port that is not member of that VID, the hints flags include this enum.</p> <p>Can be set by:</p> <p>Luton26: Y Jaguar1: Y Serval: Y Jaguar2: Y Serval2: Y ServalT: Y</p>

Definition at line 915 of file vtss_packet_api.h.

8.28.3.7 vtss_packet_tx_vstax_t

```
enum vtss_packet_tx_vstax_t
```

Transmit frames with VStaX header, and if so, how is it specified?

Enumerator

VTSS_PACKET_TX_VSTAX_NONE	Don't send frame with VStaX header.
VTSS_PACKET_TX_VSTAX_BIN	Send frame with VStaX header. The header is already encoded into binary format.
VTSS_PACKET_TX_VSTAX_SYM	Send frame with VStaX header. The header is in symbolic format and needs to be encoded by API.

Definition at line 1439 of file vtss_packet_api.h.

8.28.4 Function Documentation

8.28.4.1 vtss_npi_conf_get()

```
vtss_rc vtss_npi_conf_get (
    const vtss_inst_t inst,
    vtss_npi_conf_t *const conf )
```

Get NPI configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>conf</i>	[OUT] NPI port configuration.

Returns

Return code.

8.28.4.2 vtss_npi_conf_set()

```
vtss_rc vtss_npi_conf_set (
    const vtss_inst_t inst,
    const vtss_npi_conf_t *const conf )
```

Set NPI configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>conf</i>	[IN] NPI port configuration.

Returns

Return code.

8.28.4.3 vtss_packet_rx_conf_get()

```
vtss_rc vtss_packet_rx_conf_get (
    const vtss_inst_t inst,
    vtss_packet_rx_conf_t *const conf )
```

Get Packet Rx configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>conf</i>	[IN] Packet Rx configuration.

Returns

Return code.

8.28.4.4 vtss_packet_rx_conf_set()

```
vtss_rc vtss_packet_rx_conf_set (
    const vtss_inst_t inst,
    const vtss_packet_rx_conf_t *const conf )
```

Set CPU Rx queue configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>conf</i>	[IN] CPU Rx queue configuration.

Returns

Return code.

8.28.4.5 vtss_packet_rx_port_conf_get()

```
vtss_rc vtss_packet_rx_port_conf_get (
    const vtss_inst_t inst,
```

```
const vtss_port_no_t port_no,  
vtss_packet_rx_port_conf_t *const conf )
```

Get packet configuration for port.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>conf</i>	[OUT] Packet port configuration structure.

Returns

Return code.

8.28.4.6 vtss_packet_rx_port_conf_set()

```
vtss_rc vtss_packet_rx_port_conf_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_packet_rx_port_conf_t *const conf )
```

Set packet configuration for port.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>conf</i>	[IN] Packet port configuration structure.

Returns

Return code.

8.28.4.7 vtss_packet_rx_frame_get()

```
vtss_rc vtss_packet_rx_frame_get (
    const vtss_inst_t inst,
    const vtss_packet_rx_queue_t queue_no,
    vtss_packet_rx_header_t *const header,
    u8 *const frame,
    const u32 length )
```

Copy a received frame from a CPU queue.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>queue_no</i>	[IN] CPU queue number.
<i>header</i>	[OUT] Frame header.
<i>frame</i>	[OUT] Frame buffer.
<i>length</i>	[IN] Length of frame buffer.

Note

Depending on chipset, *queue_no* may be *don't care*. Actual queue(s) received on are returned in *header->queue_mask* as a bitmask. (I.e. frames may be received on more than one queue at a time - a '1' indicates the frame was copied for the queue at the corresponding bit position).

Returns

Return code.

8.28.4.8 vtss_packet_rx_frame_get_raw()

```
vtss_rc vtss_packet_rx_frame_get_raw (
    const vtss_inst_t inst,
    u8 *const data,
    const u32 buflen,
    u32 *const ifhlen,
    u32 *const frmlen )
```

Copy a received frame from a CPU queue - with IFH.

The extracted frame will be preceded with an IFH and will have the frame FCS at the end. The length of the IFH and frame are returned separately, but are sequentially placed in the same output buffer.

Use [vtss_packet_rx_hdr_decode\(\)](#) to decode the IFH if necessary.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>data</i>	[IN] Data buffer.
<i>buflen</i>	[IN] Length of data buffer.
<i>ifhlen</i>	[OUT] Length of IFH at the start of the buffer.
<i>frmlen</i>	[OUT] Length of received frame data - incl FCS.

Note

If the system has more than one CPU queue, a frame from the CPU queue with the lowest numerical number will be returned first. The actual queue can be decoded from the IFH.

Returns

VTSS_RC_OK if a frame was extracted, VTSS_RC_INCOMPLETE otherwise.

8.28.4.9 vtss_packet_rx_frame_discard()

```
vtss_rc vtss_packet_rx_frame_discard (
    const vtss_inst_t inst,
    const vtss_packet_rx_queue_t queue_no )
```

Discard a received frame from a CPU queue.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>queue_no</i>	[IN] CPU queue number.

Returns

Return code.

8.28.4.10 vtss_packet_tx_frame_port()

```
vtss_rc vtss_packet_tx_frame_port (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const u8 *const frame,
    const u32 length )
```

Send frame unmodified on port.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>frame</i>	[IN] Frame buffer excluding room for CRC.
<i>length</i>	[IN] Frame length excluding CRC.

Returns

Return code.

8.28.4.11 vtss_packet_tx_frame_port_vlan()

```
vtss_rc vtss_packet_tx_frame_port_vlan (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_vid_t vid,
    const u8 *const frame,
    const u32 length )
```

Send frame on port using egress rules.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>vid</i>	[IN] VLAN ID inserted if the frame is tagged on egress.
<i>frame</i>	[IN] Frame buffer excluding room for CRC.
<i>length</i>	[IN] Frame length excluding CRC.

Returns

Return code.

8.28.4.12 vtss_packet_tx_frame_vlan()

```
vtss_rc vtss_packet_tx_frame_vlan (
    const vtss_inst_t inst,
    const vtss_vid_t vid,
    const u8 *const frame,
    const u32 length )
```

Send frame on VLAN using egress rules.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>vid</i>	[IN] VLAN ID inserted if the frame is tagged on egress.
<i>frame</i>	[IN] Frame buffer excluding room for CRC.
<i>length</i>	[IN] Frame length excluding CRC.

Returns

Return code.

8.28.4.13 vtss_packet_frame_info_init()

```
void vtss_packet_frame_info_init (
    vtss_packet_frame_info_t *const info )
```

Initialize filter information to default values.

Parameters

<i>info</i>	[OUT] Frame information.
-------------	--------------------------

8.28.4.14 vtss_packet_frame_filter()

```
vtss_rc vtss_packet_frame_filter (
    const vtss_inst_t inst,
    const vtss_packet_frame_info_t *const info,
    vtss_packet_filter_t *const filter )
```

Determine ingress/egress filter for frame. The function may be used for ingress filtering, egress filtering or both.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>info</i>	[IN] Frame information.
<i>filter</i>	[OUT] Frame filter.

Returns

Return code.

8.28.4.15 vtss_packet_port_info_init()

```
vtss_rc vtss_packet_port_info_init (
    vtss_packet_port_info_t *const info )
```

Initialize filter information to default values.

Parameters

<i>info</i>	[OUT] Frame information.
-------------	--------------------------

Returns

Return code.

8.28.4.16 vtss_packet_port_filter_get()

```
vtss_rc vtss_packet_port_filter_get (
    const vtss_inst_t inst,
    const vtss_packet_port_info_t *const info,
    vtss_packet_port_filter_t filter[VTSS_PORT_ARRAY_SIZE] )
```

Determine ingress/egress filter for frame per port. The function may be used for ingress filtering, egress filtering or both.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>info</i>	[IN] Frame information.
<i>filter</i>	[OUT] Frame filter.

Returns

Return code.

8.28.4.17 vtss_packet_tx_frame_vstax()

```
vtss_rc vtss_packet_tx_frame_vstax (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_vstax_tx_header_t *const header,
    const u8 *const frame,
    const u32 length )
```

Send frame on VStaX port.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>header</i>	[IN] VStaX header structure.
<i>frame</i>	[IN] Frame buffer excluding room for CRC.
<i>length</i>	[IN] Frame length excluding CRC.

Returns

Return code.

8.28.4.18 vtss_packet_vstax_header2frame()

```
vtss_rc vtss_packet_vstax_header2frame (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_vstax_tx_header_t *const header,
    u8 *const frame )
```

Build 12 bytes VStaX frame header.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>header</i>	[IN] VStaX header structure.
<i>frame</i>	[OUT] Pointer where to write 12 bytes header in frame buffer.

Returns

Return code.

8.28.4.19 vtss_packet_vstax_frame2header()

```
vtss_rc vtss_packet_vstax_frame2header (
    const vtss_inst_t inst,
    const u8 *const frame,
    vtss_vstax_rx_header_t *const header )
```

Extract 12 bytes VStaX header.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>frame</i>	[IN] Pointer where to extract 12 bytes header from frame buffer.
<i>header</i>	[OUT] VStaX header structure.

Returns

Return code.

8.28.4.20 vtss_packet_rx_hdr_decode()

```
vtss_rc vtss_packet_rx_hdr_decode (
    const vtss_inst_t inst,
    const vtss_packet_rx_meta_t *const meta,
    const u8 hdr[VTSS_PACKET_HDR_SIZE_BYTES],
    vtss_packet_rx_info_t *const info )
```

Decode binary extraction/Rx header.

This function is mainly useful for external CPUs that wish to decode the side-band information they get on NPI ports with extraction headers enabled.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>meta</i>	[IN] Meta info on received frame.
<i>hdr</i>	[IN] Packet header (IFH)
<i>info</i>	[OUT] Decoded extraction header.

Returns

Return code. On some architectures, it is possible to detect whether the function is invoked with a valid binary extraction header, in which case the function can return an error if it's not valid. On others it can't, in which case the return code can only be different from VTSS_RC_OK if called with invalid arguments like NULL-pointers.

8.28.4.21 vtss_packet_tx_hdr_encode()

```
vtss_rc vtss_packet_tx_hdr_encode (
    const vtss_inst_t inst,
    const vtss_packet_tx_info_t *const info,
    u8 *const bin_hdr,
    u32 *const bin_hdr_len )
```

Compose binary injection/Tx header.

This function is mainly useful for external CPUs that inject frames into the switch core with an injection frame header.

In a few cases, it may also be useful for an application running on the internal CPU: If the frame is to be looped internally on a loop port, which is set-up to accept an injection header, the injection header may be encoded with this function.

The info structure is the input to the encoding, which results in a binary injection header and a length.

On many architectures, the resulting binary length is constant, but on some, it may vary with the contents of info properties. To overcome this, call this function twice. The first time, use a NULL pointer for bin_hdr. On return, the length parameter will contain the number of bytes required in bin_hdr. The second time, provide a non-NULL pointer to bin_hdr. On successful exit, bin_hdr_len will always be updated to contain the actual number of bytes required to hold the IFH. If you don't want to call this function twice, you may allocate an array of VTSS_PACKET_HDR_SIZE_BYTES (or VTSS_arch_PACKET_HDR_SIZE_BYTES) bytes, which will be long enough to accommodate all combinations of content of info. But notice that there is no guarantee that the actual length of the returned data matches VTSS_arch_PACKET_HDR_SIZE_BYTES.

For constellations that use the internal CPU in conjunction with the Frame DMA, this function will be called by the FDMA driver. Please refer to the FDMA API for details on how to specify injection properties for use with the FDMA.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>info</i>	[IN] Tx header properties.
<i>bin_hdr</i>	[OUT] NULL to get bin_hdr_len filled with required length in bytes. Non-NULL to get it filled in with the binary injection header.
<i>bin_hdr_len</i>	[INOUT] If bin_hdr is NULL, the [IN]-part is not used. Instead, it will be filled with resulting length of binary injection header in bytes. If bin_hdr is non-NULL, bin_hdr_len specifies the number of bytes that the function may write from the beginning of bin_hdr. On exit, it will contain the actual number of bytes.

Returns

Return code.

8.28.4.22 vtss_packet_tx_hdr_compile()

```
vtss_rc vtss_packet_tx_hdr_compile (
    const vtss_inst_t inst,
    const vtss_packet_tx_info_t *const info,
    vtss_packet_tx_ifh_t *const ifh )
```

Compile Tx Frame Header.

Compile a Tx frame header suitable for use with [vtss_packet_tx_frame\(\)](#).

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>info</i>	[IN] Tx header properties.
<i>ifh</i>	[OUT] Compiled Tx header.

Returns

Return code.

8.28.4.23 vtss_packet_tx_frame()

```
vtss_rc vtss_packet_tx_frame (
    const vtss_inst_t inst,
    const vtss_packet_tx_ifh_t *const ifh,
    const u8 *const frame,
    const u32 length )
```

Send frame unmodified on port with pre-compiled IFH.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>ifh</i>	[IN] Compiled IFH
<i>frame</i>	[IN] Frame buffer excluding room for CRC.
<i>length</i>	[IN] Frame length excluding CRC.

Returns

Return code.

8.28.4.24 vtss_packet_tx_info_init()

```
vtss_rc vtss_packet_tx_info_init (
    const vtss_inst_t inst,
    vtss_packet_tx_info_t *const info )
```

Initialize a Tx info structure.

Initialize the contents of a [vtss_packet_tx_info_t](#) structure.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>info</i>	[OUT] Pointer to structure that gets initialized to defaults.

Returns

VTSS_RC_OK. VTSS_RC_ERROR only if info == NULL.

8.28.4.25 vtss_packet_dma_conf_get()

```
vtss_rc vtss_packet_dma_conf_get (
    const vtss_inst_t inst,
    vtss_packet_dma_conf_t *const conf )
```

Retreive packet DMA configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>conf</i>	[OUT] Packet DMA configuration structure.

Returns

VTSS_RC_OK.

8.28.4.26 vtss_packet_dma_conf_set()

```
vtss_rc vtss_packet_dma_conf_set (
    const vtss_inst_t inst,
    const vtss_packet_dma_conf_t *const conf )
```

Set packet DMA configuration.

Zero or more packet queues may be enabled for DMA extraction.

If a queue is enabled for FDMA, it can be extracted/injected on the DMA interface of the system. Otherwise, the queues are enabled for register-based extraction/injection.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>conf</i>	[OUT] Packet DMA configuration structure.

Returns

VTSS_RC_OK.

8.28.4.27 vtss_packet_dma_offset()

```
vtss_rc vtss_packet_dma_offset (
    const vtss_inst_t inst,
    BOOL extraction,
    u32 * offset )
```

Retrive the register offset for extraction/injection DMA.

One or more queues should be enabled for DMA before using the register offset.

The returned offset is for the *status* register, which is the last register location in the DMA window. (Whole) Data word can be written by selecting the appropriate register offsets before this offset, such that the status offset is the last word written/read.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>extraction</i>	[IN]
<i>offset</i>	[OUT] Offset (32-bit word offset) for the DMA status register.

Returns

Return code.

8.29 vtss_api/include/vtss_pcs_10gbase_r_api.h File Reference

PCS_10BASE_R API.

```
#include <vtss/api/types.h>
```

8.29.1 Detailed Description

PCS_10BASE_R API.

8.30 vtss_api/include/vtss_phy_10g_api.h File Reference

10G PHY API

```
#include <vtss/api/types.h>
#include <vtss_misc_api.h>
```

Data Structures

- struct [vtss_sublayer_status_t](#)
10G Phy link and fault status
- struct [vtss_phy_10g_polarity_inv_t](#)
10G Phy Polarity inversion
- struct [vtss_phy_10g_clk_src_t](#)
10G Phy CLOCK Source Selection
- struct [ib_par_cfg](#)
Generalized data structure for IB parameters.
- struct [vtss_phy_10g_ib_conf_t](#)
10G Phy IB configuration
- struct [vtss_phy_10g_ib_status_t](#)
10G Phy IB configuration
- struct [vtss_phy_10g_apc_conf_t](#)
10G Phy APC configuration
- struct [vtss_phy_10g_apc_status_t](#)
10G Phy APC status
- struct [vtss_phy_10g_serdes_status_t](#)
10G Phy SERDES status
- struct [vtss_phy_10g_jitter_conf_t](#)
10G Phy Optimisation of jitter performance
- struct [vtss_phy_10g_mode_t](#)
10G Phy operating mode
- struct [vtss_phy_10g_init_parm_t](#)
10G Phy Initialization configuration
- struct [vtss_phy_10g_rxckout_conf_t](#)
10G Phy RXCKOUT config data
- struct [vtss_phy_10g_txckout_conf_t](#)
10G Phy TXCKOUT config data
- struct [vtss_phy_10g_srefclk_mode_t](#)
10G Phy srefclk config data
- struct [vtss_phy_10g_ckout_conf_t](#)
10G Phy CKOUT config data
- struct [vtss_phy_10g_sckout_conf_t](#)
10G Phy SCKOUT config data
- struct [vtss_phy_10g_line_clk_conf_t](#)
10G Phy Line clock config data
- struct [vtss_phy_10g_host_clk_conf_t](#)
10G Phy Host clock config data
- struct [vtss_phy_10g_lane_sync_conf_t](#)
10G Phy Lane SYNC Configuration
- struct [vtss_phy_10g_ob_status_t](#)
10G Phy OB status
- struct [vtss_phy_10g_base_kr_conf_t](#)
10G Phy 10f_base_kr_conf config data according to 802.3-2008 clause 72.7 Figure 72-11
- struct [vtss_phy_10g_base_kr_autoneg_t](#)
10G Phy Base KR Autoneg config
- struct [vtss_phy_10g_base_kr_training_t](#)
10G Phy Base KR Training config
- struct [vtss_phy_10g_base_kr_id_adv_abil_t](#)

- struct [vtss_phy_10g_base_kr_train_aneg_t](#)

10G Phy Base Link Advertisement capability config
- struct [vtss_phy_10g_kr_status_aneg_t](#)

10G Phy Base KR Training & Autoneg config
- struct [vtss_phy_10g_kr_status_train_t](#)

10G Phy Base KR Autoneg status
- struct [vtss_phy_10g_kr_status_fec_t](#)

10G Phy Base KR FEC status
- struct [vtss_phy_10g_base_kr_status_t](#)

10G Phy Base KR Training & Autoneg status
- struct [vtss_phy_10g_status_t](#)

10G Phy link and fault status for all sublayers
- struct [vtss_phy_10g_clause_37_adv_t](#)

Advertisement control data for Clause 37 aneg.
- struct [vtss_phy_10g_clause_37_status_t](#)

Clause 37 Auto-negotiation status.
- struct [vtss_phy_10g_clause_37_cmn_status_t](#)

Clause 37 Auto-negotiation status for line and host.
- struct [vtss_phy_10g_clause_37_control_t](#)

Clause 37 control struct.
- struct [vtss_phy_10g_loopback_t](#)

10G Phy system and network loopbacks
- struct [vtss_phy_pcs_cnt_t](#)

10G Phy PCS counters
- struct [vtss_phy_10g_cnt_t](#)

10G Phy Sublayer counters
- struct [vtss_phy_10g_auto_failover_conf_t](#)

10G PHY Automatic Failover configuration
- struct [vtss_phy_10g_vscope_conf_t](#)

VSCOPE fast scan storage.
- struct [vtss_phy_10g_vscope_scan_conf_t](#)

VSCOPE scan configuration.
- struct [vtss_phy_10g_vscope_scan_status_t](#)

10G Phy VSCOPE scan status
- struct [vtss_phy_10g_pcs_prbs_gen_conf_t](#)

10G Phy prbs generator Configuration
- struct [vtss_phy_10g_pcs_prbs_mon_conf_t](#)

10G Phy prbs monitor Configuration
- struct [vtss_phy_10g_prbs_gen_conf_t](#)

10G PHY prbs monitor Configuration
- struct [vtss_phy_10g_prbs_mon_conf_t](#)

10G PHY prbs monitor Configuration
- struct [vtss_phy_10g_pkt_gen_conf_t](#)

10G PHY Packet generator configuration
- struct [vtss_phy_10g_pkt_mon_conf_t](#)

10G PHY Packet Monitor configuration
- struct [vtss_phy_10g_timestamp_val_t](#)

10G PHY timestamp value array(holder)
- struct [vtss_phy_10g_id_t](#)

10G Phy part number and revision
- struct [vtss_gpio_10g_gpio_mode_t](#)

GPIO configured mode.
- struct [vtss_phy_10g_fw_status_t](#)

Firmware status.

Macros

- #define VTSS_SLEWRATE_25PS 0
Slew rate ctrl of OB.
- #define VTSS_SLEWRATE_35PS 1
- #define VTSS_SLEWRATE_55PS 2
- #define VTSS_SLEWRATE_70PS 3
- #define VTSS_SLEWRATE_120PS 4
- #define VTSS_SLEWRATE_INVALID 5
- #define BOOLEAN_STORAGE_COUNT 6
- #define UNSIGNED_STORAGE_COUNT 5
- #define PHASE_POINTS 128
- #define AMPLITUDE_POINTS 64
- #define VTSS_PHY_10G_ONE_LINE_ACTIVE 0x08
- #define VTSS_PHY_10G_MACSEC_DISABLED 0x04
- #define VTSS_PHY_10G_TIMESTAMP_DISABLED 0x02
- #define VTSS_PHY_10G_MACSEC_KEY_128 0x01
- #define VTSS_10G_PHY_GPIO_MAX 12
- #define VTSS_10G_PHY_GPIO_MAL_MAX 40
- #define VTSS_PHY_10G_LINK_LOS_EV 0x00000001
Event source identification mask values.
- #define VTSS_PHY_10G_RX_LOL_EV 0x00000002
- #define VTSS_PHY_10G_TX_LOL_EV 0x00000004
- #define VTSS_PHY_10G_LOPC_EV 0x00000008
- #define VTSS_PHY_10G_HIGH_BER_EV 0x00000010
- #define VTSS_PHY_10G_MODULE_STAT_EV 0x00000020
- #define VTSS_PHY_10G_PCS_RECEIVE_FAULT_EV 0x00000040
- #define VTSS_PHY_EWIS_SEF_EV 0x00000080
- #define VTSS_PHY_EWIS_FPLM_EV 0x00000100
- #define VTSS_PHY_EWIS_FAIS_EV 0x00000200
- #define VTSS_PHY_EWIS_LOF_EV 0x00000400
- #define VTSS_PHY_EWIS_RDIL_EV 0x00000800
- #define VTSS_PHY_EWIS_AISL_EV 0x00001000
- #define VTSS_PHY_EWIS_LCDP_EV 0x00002000
- #define VTSS_PHY_EWIS_PLMP_EV 0x00004000
- #define VTSS_PHY_EWIS_AISP_EV 0x00008000
- #define VTSS_PHY_EWIS_LOPP_EV 0x00010000
- #define VTSS_PHY_EWIS_UNEQP_EV 0x00020000
- #define VTSS_PHY_EWIS_FEUNEQP_EV 0x00040000
- #define VTSS_PHY_EWIS_FERDIP_EV 0x00080000
- #define VTSS_PHY_EWIS_REIL_EV 0x00100000
- #define VTSS_PHY_EWIS_REIP_EV 0x00200000
- #define VTSS_PHY_EWIS_B1_NZ_EV 0x00400000
- #define VTSS_PHY_EWIS_B2_NZ_EV 0x00800000
- #define VTSS_PHY_EWIS_B3_NZ_EV 0x01000000
- #define VTSS_PHY_EWIS_REIL_NZ_EV 0x02000000
- #define VTSS_PHY_EWIS_REIP_NZ_EV 0x04000000
- #define VTSS_PHY_EWIS_B1_THRESH_EV 0x08000000
- #define VTSS_PHY_EWIS_B2_THRESH_EV 0x10000000
- #define VTSS_PHY_EWIS_B3_THRESH_EV 0x20000000
- #define VTSS_PHY_EWIS_REIL_THRESH_EV 0x40000000
- #define VTSS_PHY_EWIS_REIP_THRESH_EV 0x80000000
- #define VTSS_PHY_10G_RX_LOS_EV 0x00000001
- #define VTSS_PHY_10G_RX_LOL_EV 0x00000002

- #define VTSS_PHY_10G_TX_LOL_EV 0x00000004
- #define VTSS_PHY_10G_RX_CHAR_DEC_CNT_THRESH_EV 0x00000010
- #define VTSS_PHY_10G_RX_CHAR_ENC_CNT_THRESH_EV 0x00000020
- #define VTSS_PHY_10G_RX_BLK_DEC_CNT_THRESH_EV 0x00000040
- #define VTSS_PHY_10G_RX_BLK_ENC_CNT_THRESH_EV 0x00000080
- #define VTSS_PHY_10G_RX_SEQ_CNT_THRESH_EV 0x00000100
- #define VTSS_PHY_10G_RX_SEQ_CNT_THRESH_EV 0x00000200
- #define VTSS_PHY_10G_FEC_UNFIXED_CNT_THRESH_EV 0x00000400
- #define VTSS_PHY_10G_FEC_FIXED_CNT_THRESH_EV 0x00000800
- #define VTSS_PHY_10G_HIGHBER_EV 0x00001000
- #define VTSS_PHY_10G_RX_LINK_STAT_EV 0x00002000
- #define VTSS_PHY_10G_RX_LINK_STAT_EV 0x00002000
- #define VTSS_PHY_10G_GPIO_INT_AGG0_EV 0x00004000
- #define VTSS_PHY_10G_GPIO_INT_AGG1_EV 0x00008000
- #define VTSS_PHY_10G_GPIO_INT_AGG2_EV 0x00010000
- #define VTSS_PHY_10G_GPIO_INT_AGG3_EV 0x00020000
- #define VTSS_PHY_1G_LINE_AUTONEG_RESTART_EV 0x00040000
- #define VTSS_PHY_1G_HOST_AUTONEG_RESTART_EV 0x00080000
- #define VTSS_PHY_10G_LINE_MAC_LOCAL_FAULT_EV 0x00100000
- #define VTSS_PHY_10G_HOST_MAC_LOCAL_FAULT_EV 0x00400000
- #define VTSS_PHY_10G_LINE_MAC_REMOTE_FAULT_EV 0x00800000
- #define VTSS_PHY_10G_HOST_MAC_REMOTE_FAULT_EV 0x01000000

Typedefs

- typedef u16 vtss_gpio_10g_no_t
- typedef enum ckout_sel_ckout_sel_t

10G Phy CKOUTs Enum
- typedef u32 vtss_32_cntr_t
- typedef u32 vtss_phy_10g_event_t
- typedef u32 vtss_phy_10g_extnd_event_t

Enumerations

- enum oper_mode_t {
 VTSS_PHY_LAN_MODE, VTSS_PHY_WAN_MODE, VTSS_PHY_1G_MODE, VTSS_PHY_LAN_SYNCE_MODE,
 VTSS_PHY_WAN_SYNCE_MODE, VTSS_PHY_LAN_MIXED_SYNCE_MODE, VTSS_PHY_WAN_MIXED_SYNCE_MODE,
 VTSS_PHY_REPEATERO_MODE }

10G Phy operating mode enum type
- enum vtss_wrefclk_t { VTSS_WREFCLK_155_52, VTSS_WREFCLK_622_08 }

Modes for WAN reference clock.
- enum vtss_phy_interface_mode {
 VTSS_PHY_XAUI_XFI, VTSS_PHY_XGMII_XFI, VTSS_PHY_RXAUI_XFI, VTSS_PHY_SGMII_LANE_0_XFI,
 VTSS_PHY_SGMII_LANE_3_XFI, VTSS_PHY_SFI_XFI }

Phy Interface modes.
- enum vtss_recvrd_t { VTSS_RECVRD_RXCLKOUT, VTSS_RECVRD_TXCLKOUT }

Modes for recovered clock.
- enum vtss_recvrdclk_cdr_div_t { VTSS_RECVRDCLK_CDR_DIV_64, VTSS_RECVRDCLK_CDR_DIV_66 }

Modes for recovered clock divisor.
- enum vtss_srefclk_div_t { VTSS_SREFCLK_DIV_64, VTSS_SREFCLK_DIV_66, VTSS_SREFCLK_DIV_16 }

Modes for Synch-E recovered clock.

- enum `vtss_wref_clk_div_t` { `VTSS_WREFCLK_NONE`, `VTSS_WREFCLK_DIV_16` }

Modes for WREFCLK clock divisor.
- enum `apc_ib_regulator_t` { `VTSS_AP_C_IB_SFP_PLUS_ZR`, `VTSS_AP_C_IB_BACKPLANE` }

APC Rx regulator mode.
- enum `ddr_mode_t` { `VTSS_DDR_MODE_A`, `VTSS_DDR_MODE_K`, `VTSS_DDR_MODE_M` }

Interleave mode.
- enum `clk_mstr_t` { `VTSS_CLK_MSTR_INTERNAL`, `VTSS_CLK_MSTR_EXTERNAL` }

Clock master.
- enum `vtss_rptr_rate_t` {
 `VTSS_RPTR_RATE_NONE`, `VTSS_RPTR_RATE_10_3125`, `VTSS_RPTR_RATE_9_9532`, `VTSS_RPTR_RATE_11_3`,
`VTSS_RPTR_RATE_10_5187`, `VTSS_RPTR_RATE_1_25`, `VTSS_RPTR_RATE_10_709`, `VTSS_RPTR_RATE_11_095727`,
`VTSS_RPTR_RATE_11_05` }

Repeater Data rate.
- enum `vtss_phy_10g_media_t` {
 `VTSS_MEDIA_TYPE_SR`, `VTSS_MEDIA_TYPE_SR2`, `VTSS_MEDIA_TYPE_DAC`, `VTSS_MEDIA_TYPE_ZR`,
`VTSS_MEDIA_TYPE_KR`, `VTSS_MEDIA_TYPE_SR_SC`, `VTSS_MEDIA_TYPE_SR2_SC`, `VTSS_MEDIA_TYPE_DAC_SC`,
`VTSS_MEDIA_TYPE_ZR_SC`, `VTSS_MEDIA_TYPE_ZR2_SC`, `VTSS_MEDIA_TYPE_KR_SC`, `VTSS_MEDIA_TYPE_NONE` }

10G Phy Media type
- enum `vtss_phy_6g_link_partner_distance_t` { `VTSS_6G_LINK_SHORT_RANGE`, `VTSS_6G_LINK_LONG_RANGE` }

6G serdes link partner distance selection
- enum `vtss_phy_10g_ib_apc_op_mode_t` {
 `VTSS_IB_AP_C_AUTO`, `VTSS_IB_AP_C_MANUAL`, `VTSS_IB_AP_C_FREEZE`, `VTSS_IB_AP_C_RESET`,
`VTSS_IB_AP_C_RESTART`, `VTSS_IB_AP_C_NONE` }

10G SERDES APC operation
- enum `vtss_channel_t` {
 `VTSS_CHANNEL_AUTO`, `VTSS_CHANNEL_0`, `VTSS_CHANNEL_1`, `VTSS_CHANNEL_2`,
`VTSS_CHANNEL_3` }

Channel modes - Auto is recommended.
- enum `vtss_reccrd_clkout_t` { `VTSS_RECVRD_CLKOUT_DISABLE`, `VTSS_RECVRD_CLKOUT_LINE_SIDE_RX_CLOCK`,
`VTSS_RECVRD_CLKOUT_LINE_SIDE_TX_CLOCK` }

Modes for (rx/tx) recovered clock output.
- enum `vtss_phy_10g_srefclk_freq_t` { `VTSS_PHY_10G_SREFCLK_156_25`, `VTSS_PHY_10G_SREFCLK_125_00`,
`VTSS_PHY_10G_SREFCLK_155_52`, `VTSS_PHY_10G_SREFCLK_INVALID` }

10G Phy sref clock input frequency
- enum `vtss_phy_10g_ckout_freq_t` { `VTSS_PHY_10G_CLK_FULL_RATE`, `VTSS_PHY_10G_CLK_DIVIDE_BY_2`,
`VTSS_PHY_10G_CLK_INVALID` }

10G Phy clock frequency
- enum `vtss_ckout_data_sel_t` {
 `VTSS_CKOUT_LINE0_TX_CLOCK`, `VTSS_CKOUT_LINE1_TX_CLOCK`, `VTSS_CKOUT_LINE2_TX_CLOCK`,
`VTSS_CKOUT_LINE3_TX_CLOCK`,
`VTSS_CKOUT_HOST0_TX_CLOCK`, `VTSS_CKOUT_HOST1_TX_CLOCK`, `VTSS_CKOUT_HOST2_TX_CLOCK`,
`VTSS_CKOUT_HOST3_TX_CLOCK`,
`VTSS_CKOUT_LINE0_RECVRD_CLOCK`, `VTSS_CKOUT_LINE1_RECVRD_CLOCK`, `VTSS_CKOUT_LINE2_RECVRD_CLOCK`,
`VTSS_CKOUT_LINE3_RECVRD_CLOCK`,
`VTSS_CKOUT_HOST0_RECVRD_CLOCK`, `VTSS_CKOUT_HOST1_RECVRD_CLOCK`, `VTSS_CKOUT_HOST2_RECVRD_CLOCK`,
`VTSS_CKOUT_HOST3_RECVRD_CLOCK`,
`VTSS_CKOUT_HOST_PLL_CLOCK`, `VTSS_CKOUT_LINE_PLL_CLOCK`, `VTSS_CKOUT_CSR_CLOCK`,
`VTSS_CKOUT_LTC_CLOCK`,
`VTSS_CKOUT_DF2F_CLOCK`, `VTSS_CKOUT_F2DF_CLOCK`, `VTSS_CKOUT_DEBUG1`, `VTSS_CKOUT_DEBUG2`,
`VTSS_CKOUT_OSCILLATOR_OUTPUT` }

Modes for recovered clock output.

- enum `vtss_phy_10g_squelch_src_t` {

VTSS_CKOUT_SQUELCH_SRC_GPIO0, VTSS_CKOUT_SQUELCH_SRC_GPIO1, VTSS_CKOUT_SQUELCH_SRC_GPIO2,

VTSS_CKOUT_SQUELCH_SRC_GPIO3,

VTSS_CKOUT_SQUELCH_SRC_GPIO4, VTSS_CKOUT_SQUELCH_SRC_GPIO5, VTSS_CKOUT_SQUELCH_SRC_GPIO6,

VTSS_CKOUT_SQUELCH_SRC_GPIO7,

VTSS_CKOUT_SQUELCH_SRC_LINK_LINE0, VTSS_CKOUT_SQUELCH_SRC_LINK_LINE1, VTSS_CKOUT_SQUELCH_SRC_LINK_LINE2,

VTSS_CKOUT_SQUELCH_SRC_LINK_LINE3,

VTSS_CKOUT_SQUELCH_SRC_LINK_HOST0, VTSS_CKOUT_SQUELCH_SRC_LINK_HOST1, VTSS_CKOUT_SQUELCH_SRC_LINK_HOST2,

VTSS_CKOUT_SQUELCH_SRC_LINK_HOST3,

VTSS_CKOUT_SQUELCH_SRC_LOS_LINE0, VTSS_CKOUT_SQUELCH_SRC_LOS_LINE1, VTSS_CKOUT_SQUELCH_SRC_LOS_LINE2,

VTSS_CKOUT_SQUELCH_SRC_LOS_LINE3,

VTSS_CKOUT_SQUELCH_SRC_LOS_HOST0, VTSS_CKOUT_SQUELCH_SRC_LOS_HOST1, VTSS_CKOUT_SQUELCH_SRC_LOS_HOST2,

VTSS_CKOUT_SQUELCH_SRC_LOS_HOST3,

VTSS_CKOUT_SQUELCH_SRC_LINK_LINE0_KR, VTSS_CKOUT_SQUELCH_SRC_LINK_LINE1_KR,

VTSS_CKOUT_SQUELCH_SRC_LINK_LINE2_KR, VTSS_CKOUT_SQUELCH_SRC_LINK_LINE3_KR,

VTSS_CKOUT_SQUELCH_SRC_LINK_HOST0_KR, VTSS_CKOUT_SQUELCH_SRC_LINK_HOST1_KR,

VTSS_CKOUT_SQUELCH_SRC_LINK_HOST2_KR, VTSS_CKOUT_SQUELCH_SRC_LINK_HOST3_KR,

VTSS_CKOUT_NO_SQUELCH }

squelch control source
- enum `vtss_phy_10g_clk_sel_t` {

VTSS_PHY_10G_LINE0_RECVRD_CLOCK, VTSS_PHY_10G_LINE1_RECVRD_CLOCK, VTSS_PHY_10G_LINE2_RECVRD_CLOCK,

VTSS_PHY_10G_LINE3_RECVRD_CLOCK,

VTSS_PHY_10G_HOST0_RECVRD_CLOCK, VTSS_PHY_10G_HOST1_RECVRD_CLOCK, VTSS_PHY_10G_HOST2_RECVRD_CLOCK,

VTSS_PHY_10G_HOST3_RECVRD_CLOCK,

VTSS_PHY_10G_SREFCLK, VTSS_PHY_10G_SYNC_DISABLE = 15 }

Modes of recovered clocks for ckout and sckout pins.
- enum `vtss_phy_10g_recvrd_clk_sel_t` {

VTSS_PHY_10G_USE_LINE0_RECVRD_CLOCK, VTSS_PHY_10G_USE_LINE1_RECVRD_CLOCK,

VTSS_PHY_10G_USE_LINE2_RECVRD_CLOCK, VTSS_PHY_10G_USE_LINE3_RECVRD_CLOCK,

VTSS_PHY_10G_USE_HOST0_RECVRD_CLOCK, VTSS_PHY_10G_USE_HOST1_RECVRD_CLOCK,

VTSS_PHY_10G_USE_HOST2_RECVRD_CLOCK, VTSS_PHY_10G_USE_HOST3_RECVRD_CLOCK,

VTSS_PHY_10G_USE_SREFCLK_CLOCK, VTSS_PHY_10G_USE_DEFAULT_RECVRD_CLOCK }

Modes of recovered clock selection.
- enum `ckout_sel_t` { **CKOUT0, CKOUT1, CKOUT2, CKOUT3** }

10G Phy CKOUTs Enum
- enum `vtss_phy_10g_sckout_freq_t` { VTSS_PHY_10G_SCKOUT_156_25, VTSS_PHY_10G_SCKOUT_125_00,

VTSS_PHY_10G_SCKOUT_INVALID }

10G Phy sckout clock input frequency
- enum `vtss_phy_10g_rx_macro_t` { VTSS_PHY_10G_RX_MACRO_LINE, VTSS_PHY_10G_RX_MACRO_HOST,

VTSS_PHY_10G_RX_MACRO_SREFCLK }

10G Phy Rx MACRO Configuration
- enum `vtss_phy_10g_tx_macro_t` { VTSS_PHY_10G_TX_MACRO_LINE, VTSS_PHY_10G_TX_MACRO_HOST,

VTSS_PHY_10G_TX_MACRO_SCKOUT }

10G Phy tx MACRO Configuration
- enum `vtss_phy_10g_clause_37_remote_fault_t` { VTSS_PHY_10G_CLAUSE_37_RF_LINK_OK, VTSS_PHY_10G_CLAUSE_37_RF_LINK_FAILURE,

VTSS_PHY_10G_CLAUSE_37_RF_LINK_FAILURE, VTSS_PHY_10G_CLAUSE_37_RF_AUTONEG_ERROR }

Auto-negotiation remote fault type.
- enum `vtss_lb_type_t` {

VTSS_LB_NONE, VTSS_LB_SYSTEM_XS_SHALLOW, VTSS_LB_SYSTEM_XS_DEEP, VTSS_LB_SYSTEM_PCS_SHALLOW,

VTSS_LB_SYSTEM_PCS_DEEP, VTSS_LB_SYSTEM_PMA, VTSS_LB_NETWORK_XS_SHALLOW,

VTSS_LB_NETWORK_XS_DEEP,

VTSS_LB_NETWORK_PCS, VTSS_LB_NETWORK_WIS, VTSS_LB_NETWORK_PMA, VTSS_LB_H2,

VTSS_LB_H3, VTSS_LB_H4, VTSS_LB_H5, VTSS_LB_H6,

VTSS_LB_L0, VTSS_LB_L1, VTSS_LB_L2, VTSS_LB_L3,

VTSS_LB_L2C }

- *10G loopback types*
- enum `vtss_phy_10g_power_t` { `VTSS_PHY_10G_POWER_ENABLE`, `VTSS_PHY_10G_POWER_DISABLE` }
- 10G Phy power setting*
- enum `vtss_phy_10g_failover_mode_t` {
`VTSS_PHY_10G_PMA_TO_FROM_XAUI_NORMAL`, `VTSS_PHY_10G_PMA_TO_FROM_XAUI_CROSSED`,
`VTSS_PHY_10G_PMA_0_TO_FROM_XAUI_0_TO_XAUI_1`, `VTSS_PHY_10G_PMA_0_TO_FROM_XAUI_1_TO_XAUI_0`,
`VTSS_PHY_10G_PMA_1_TO_FROM_XAUI_0_TO_XAUI_1`, `VTSS_PHY_10G_PMA_1_TO_FROM_XAUI_1_TO_XAUI_0` }
- 10G Phy Failover Mode Setting*
- enum `vtss_phy_10g_auto_failover_event_t` {
`VTSS_PHY_10G_AUTO_FAILOVER_EVENT_PCS_LINK_STATUS`, `VTSS_PHY_10G_AUTO_FAILOVER_EVENT_SERDES`,
`VTSS_PHY_10G_AUTO_FAILOVER_EVENT_WIS_LOF`, `VTSS_PHY_10G_AUTO_FAILOVER_EVENT_GPIO`,
`VTSS_PHY_10G_AUTO_FAILOVER_EVENT_NONE` }
- 10G Phy Automatic Failover Event Setting*
- enum `vtss_phy_10g_auto_failover_filter_t` {
`VTSS_PHY_10G_AUTO_FAILOVER_FILTER_NONE`, `VTSS_PHY_10G_AUTO_FAILOVER_FILTER_CNT_B2316`,
`VTSS_PHY_10G_AUTO_FAILOVER_FILTER_CNT_B70`, `VTSS_PHY_10G_AUTO_FAILOVER_FILTER_CNT_A2316`,
`VTSS_PHY_10G_AUTO_FAILOVER_FILTER_CNT_A70` }
- 10G PHY Automatic Failover Filter*
- enum `vtss_phy_10g_vscope_scan_t` { `VTSS_PHY_10G_FAST_SCAN`, `VTSS_PHY_10G_FAST_SCAN_PLUS`,
`VTSS_PHY_10G_QUICK_SCAN`, `VTSS_PHY_10G_FULL_SCAN` }
- VSCOPE scan types.*
- enum `vtss_phy_10g_pkt_mon_rst_t` {
`VTSS_PHY_10G_PKT_MON_RST_ALL`, `VTSS_PHY_10G_PKT_MON_RST_GOOD`, `VTSS_PHY_10G_PKT_MON_RST_BAD`,
`VTSS_PHY_10G_PKT_MON_RST_FRAG`,
`VTSS_PHY_10G_PKT_MON_RST_LFAULT`, `VTSS_PHY_10G_PKT_MON_RST_BER`, `VTSS_PHY_10G_PKT_MON_RST_NOMATCH` }
- 10G PHY Packet monitor configuration*
- enum `vtss_phy_10g_type_t` {
`VTSS_PHY_TYPE_10G_NONE` = 0, `VTSS_PHY_TYPE_8484` = 8484, `VTSS_PHY_TYPE_8486` = 8486,
`VTSS_PHY_TYPE_8487` = 8487,
`VTSS_PHY_TYPE_8488` = 8488, `VTSS_PHY_TYPE_8489` = 8489, `VTSS_PHY_TYPE_8489_15` = 848915,
`VTSS_PHY_TYPE_8490` = 8490,
`VTSS_PHY_TYPE_8491` = 8491, `VTSS_PHY_TYPE_8256` = 8256, `VTSS_PHY_TYPE_8257` = 8257, `VTSS_PHY_TYPE_8258` = 8258,
`VTSS_PHY_TYPE_8254` = 8254 }
- 10g PHY type*
- enum `vtss_phy_10g_family_t` {
`VTSS_PHY_FAMILY_10G_NONE`, `VTSS_PHY_FAMILY_XAUI_XGMII_XFI`, `VTSS_PHY_FAMILY_XAU_I_XFI`,
`VTSS_PHY_FAMILY_VENICE`,
`VTSS_PHY_FAMILY_MALIBU` }
- 10G PHY family*
- enum `vtss_10g_phy_gpio_t` {
`VTSS_10G_PHY_GPIO_NOT_INITIALIZED`, `VTSS_10G_PHY_GPIO_OUT`, `VTSS_10G_PHY_GPIO_IN`,
`VTSS_10G_PHY_GPIO_WIS_INT`,
`VTSS_10G_PHY_GPIO_1588_LOAD_SAVE`, `VTSS_10G_PHY_GPIO_1588_1PPS_0`, `VTSS_10G_PHY_GPIO_1588_1PPS_1`,
`VTSS_10G_PHY_GPIO_1588_1PPS_2`,
`VTSS_10G_PHY_GPIO_1588_1PPS_3`, `VTSS_10G_PHY_GPIO_PCS_RX_FAULT`, `VTSS_10G_PHY_GPIO_SET_I2C_MASTER`,
`VTSS_10G_PHY_GPIO_TX_ENABLE`,
`VTSS_10G_PHY_GPIO_LINE_PLL_STATUS`, `VTSS_10G_PHY_GPIO_HOST_PLL_STATUS`, `VTSS_10G_PHY_GPIO_RCO`,
`VTSS_10G_PHY_GPIO_CHAN_INT_0`,
`VTSS_10G_PHY_GPIO_CHAN_INT_1`, `VTSS_10G_PHY_GPIO_1588_INT`, `VTSS_10G_PHY_GPIO_TS_FIFO_EMPTY`,
`VTSS_10G_PHY_GPIO_AGG_INT_0`,
`VTSS_10G_PHY_GPIO_AGG_INT_1`, `VTSS_10G_PHY_GPIO_AGG_INT_2`, `VTSS_10G_PHY_GPIO_AGG_INT_3`,
`VTSS_10G_PHY_GPIO_PLL_INT_0`,

```
VTSS_10G_PHY_GPIO_PLL_INT_1, VTSS_10G_PHY_GPIO_SET_I2C_SLAVE, VTSS_10G_PHY_GPIO_CRSS_INT,
VTSS_10G_PHY_GPIO_LED,
VTSS_10G_PHY_GPIO_DRIVE_LOW, VTSS_10G_PHY_GPIO_DRIVE_HIGH }
```

GPIO configured mode.

- enum `vtss_gpio_10g_gpio_intr_sgnl_t` {


```
VTSS_10G_GPIO_INTR_SGNL_I2C_MSTR_DATA_OUT, VTSS_10G_GPIO_INTR_SGNL_I2C_MSTR_CLK_OUT,
VTSS_10G_GPIO_INTR_SGNL_LED_TX, VTSS_10G_GPIO_INTR_SGNL_LED_RX,
VTSS_10G_GPIO_INTR_SGNL_RX_ALARM, VTSS_10G_GPIO_INTR_SGNL_TX_ALARM, VTSS_10G_GPIO_INTR_SGNL_
VTSS_10G_GPIO_INTR_SGNL_LINE_LINK,
VTSS_10G_GPIO_INTR_SGNL_LINE_KR_8b10b_2GPIO, VTSS_10G_GPIO_INTR_SGNL_LINE_KR_10b_2GPIO,
VTSS_10G_GPIO_INTR_SGNL_ROSI_PULSE, VTSS_10G_GPIO_INTR_SGNL_ROSI_SDATA,
VTSS_10G_GPIO_INTR_SGNL_ROSI_SCLK, VTSS_10G_GPIO_INTR_SGNL_TOSI_PULSE, VTSS_10G_GPIO_INTR_SGNL_
VTSS_10G_GPIO_INTR_SGNL_LINE_PCS1G_LINK,
VTSS_10G_GPIO_INTR_SGNL_LINE_PCS_RX_STAT, VTSS_10G_GPIO_INTR_SGNL_CLIENT_PCS1G_LINK,
VTSS_10G_GPIO_INTR_SGNL_HOST_PCS_RX_STAT, VTSS_10G_GPIO_INTR_SGNL_HOST_SD10G_IB_SIG,
VTSS_10G_GPIO_INTR_SGNL_LINE_SD10G_IB_SIG, VTSS_10G_GPIO_INTR_SGNL_HPCS_INTR,
VTSS_10G_GPIO_INTR_SGNL_LPCS_INTR, VTSS_10G_GPIO_INTR_SGNL_CLIENT_PCS1G_INTR,
VTSS_10G_GPIO_INTR_SGNL_LINE_PCS1G_INTR, VTSS_10G_GPIO_INTR_SGNL_WIS_INT0,
VTSS_10G_GPIO_INTR_SGNL_HOST_PMA_INT, VTSS_10G_GPIO_INTR_SGNL_LINE_PMA_INT,
VTSS_10G_GPIO_INTR_SGNL_DATA_ACT_TX, VTSS_10G_GPIO_INTR_SGNL_DATA_ACT_RX,
VTSS_10G_GPIO_INTR_SGNL_HDATA_ACT_TX, VTSS_10G_GPIO_INTR_SGNL_HDATA_ACT_RX,
VTSS_10G_GPIO_INTR_SGNL_XGMII_PAUS_EGR, VTSS_10G_GPIO_INTR_SGNL_XGMII_PAUS_ING,
VTSS_10G_GPIO_INTR_SGNL_RX_PCS_PAUS, VTSS_10G_GPIO_INTR_SGNL_TX_PCS_PAUS,
VTSS_10G_GPIO_INTR_SGNL_RX_WIS_PAUS, VTSS_10G_GPIO_INTR_SGNL_TX_WIS_PAUS,
VTSS_10G_GPIO_INTR_SGNL_ETH_CHAN_DIS, VTSS_10G_GPIO_INTR_SGNL_MACSEC_1588_SFD_LANE,
VTSS_10G_GPIO_INTR_SGNL_LINE_S_TXFAULT, VTSS_10G_GPIO_INTR_SGNL_LPCS1G_LATENCY0_OR_EWIS_BIT,
VTSS_10G_GPIO_INTR_SGNL_LPCS1G_LATENCY1_OR_EWIS_BIT1, VTSS_10G_GPIO_INTR_SGNL_LPCS1G_CHAR,
VTSS_10G_GPIO_INTR_SGNL_LPCS1G_CHAR_POS1_OR_EWIS_WORD0, VTSS_10G_GPIO_INTR_SGNL_LPCS1G_CHAR,
VTSS_10G_GPIO_INTR_SGNL_LPCS1G_CHAR_POS3_OR_EWIS_WORD2, VTSS_10G_GPIO_INTR_SGNL_MACSEC_IC,
VTSS_10G_GPIO_INTR_SGNL_MACSEC_IGR_PRED_VAR1, VTSS_10G_GPIO_INTR_SGNL_KR_ACTV_2GPIO,
VTSS_10G_GPIO_INTR_SGNL_DFT_TX_2GPIO, VTSS_10G_GPIO_INTR_SGNL_RESERVED,
VTSS_10G_GPIO_INTR_SGNL_EXE_LST_2GPIO_0, VTSS_10G_GPIO_INTR_SGNL_EXE_LST_2GPIO_1,
VTSS_10G_GPIO_INTR_SGNL_EXE_LST_2GPIO_2, VTSS_10G_GPIO_INTR_SGNL_EXE_LST_2GPIO_3,
VTSS_10G_GPIO_INTR_SGNL_EXE_LST_2GPIO_4, VTSS_10G_GPIO_INTR_SGNL_LINK_HCD_2GPIO_0,
VTSS_10G_GPIO_INTR_SGNL_LINK_HCD_2GPIO_1, VTSS_10G_GPIO_INTR_SGNL_LINK_HCD_2GPIO_2,
VTSS_10G_GPIO_INTR_SGNL_ETH_1G_ENA, VTSS_10G_GPIO_INTR_SGNL_H_KR_8b10b_2GPIO,
VTSS_10G_GPIO_INTR_SGNL_H_KR_10Gb_2GPIO, VTSS_10G_GPIO_INTR_SGNL_H_KR_ACTV_2GPIO,
VTSS_10G_GPIO_INTR_SGNL_NONE }
```

GPIO internal signal types.

- enum `vtss_gpio_10g_chan_intrpt_t` {


```
VTSS_10G_GPIO_INTRPT_WIS0, VTSS_10G_GPIO_INTRPT_WIS1, VTSS_10G_GPIO_INTRPT_LPCS10G,
VTSS_10G_GPIO_INTRPT_HPCS10G,
VTSS_10G_GPIO_INTRPT_LPCS1G, VTSS_10G_GPIO_INTRPT_HPCS1G, VTSS_10G_GPIO_INTRPT_MSEC_EGR,
VTSS_10G_GPIO_INTRPT_MSEC_IGR,
VTSS_10G_GPIO_INTRPT_LMAC, VTSS_10G_GPIO_INTRPT_HMAC, VTSS_10G_GPIO_INTRPT_FCBUF,
VTSS_10G_GPIO_INTRPT_LIGR_FIFO,
VTSS_10G_GPIO_INTRPT_LEGRT_FIFO, VTSS_10G_GPIO_INTRPT_HEGR_FIFO, VTSS_10G_GPIO_INTRPT_LPMA,
VTSS_10G_GPIO_INTRPT_HPMA }
```

GPIO Channel level interrupts.

- enum `vtss_gpio_10g_aggr_intrpt_t` {


```
VTSS_10G_GPIO_AGGR_INTRPT_CH0_INTR0_EN, VTSS_10G_GPIO_AGGR_INTRPT_CH0_INTR1_EN,
VTSS_10G_GPIO_AGGR_INTRPT_CH1_INTR0_EN, VTSS_10G_GPIO_AGGR_INTRPT_CH1_INTR1_EN,
VTSS_10G_GPIO_AGGR_INTRPT_CH2_INTR0_EN, VTSS_10G_GPIO_AGGR_INTRPT_CH2_INTR1_EN,
VTSS_10G_GPIO_AGGR_INTRPT_CH3_INTR0_EN, VTSS_10G_GPIO_AGGR_INTRPT_CH3_INTR1_EN,
VTSS_10G_GPIO_AGGR_INTRPT_IP1588_0_INTR0_EN, VTSS_10G_GPIO_AGGR_INTRPT_IP1588_0_INTR1_EN,
VTSS_10G_GPIO_AGGR_INTRPT_IP1588_0_INTR2_EN, VTSS_10G_GPIO_AGGR_INTRPT_IP1588_0_INTR3_EN,
VTSS_10G_GPIO_AGGR_INTRPT_IP1588_1_INTR0_EN, VTSS_10G_GPIO_AGGR_INTRPT_IP1588_1_INTR1_EN,
VTSS_10G_GPIO_AGGR_INTRPT_IP1588_1_INTR2_EN, VTSS_10G_GPIO_AGGR_INTRPT_IP1588_1_INTR3_EN,
```

```
VTSS_10G_GPIO_AGGR_INTRPT_LCPLL_0_INTR_EN, VTSS_10G_GPIO_AGGR_INTRPT_LCPLL_1_INTR_EN,
VTSS_10G_GPIO_AGGR_INTRPT_EXP4_INTR_EN, VTSS_10G_GPIO_AGGR_INTRPT_CLK_MUX_INTR_EN,
VTSS_10G_GPIO_AGGR_INTRPT_GPIO_INTR_EN }
```

GPIO Channel level interrupts.

- enum `vtss_gpio_10g_input_t` { `VTSS_10G_GPIO_INPUT_NONE`, `VTSS_10G_GPIO_INPUT_LINE_LOPC`,
`VTSS_10G_GPIO_INPUT_HOST_LOPC` }

GPIO Channel level interrupts.

Functions

- `vtss_rc vtss_phy_10g_mode_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_10g_mode_t` *const mode)
Get the Phy operating mode.
- `vtss_rc vtss_phy_10g_init` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_phy_10g_init_parm_t` *const init_conf)
Identify PHY and initialize software accordingly.
- `vtss_rc vtss_phy_10g_mode_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_phy_10g_mode_t` *const mode)
Identify, Reset and set the operating mode of the PHY.
- `vtss_rc vtss_phy_10g_ib_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_phy_10g_ib_conf_t` *const ib_conf, `BOOL` is_host)
Configure Input buffer .
- `vtss_rc vtss_phy_10g_ib_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `BOOL` is_host, `vtss_phy_10g_ib_conf_t` *const ib_conf)
Get configuration of Input buffer .
- `vtss_rc vtss_phy_10g_ib_status_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_10g_ib_status_t` *const ib_status)
Get status of Input buffer .
- `vtss_rc vtss_phy_10g_apc_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_phy_10g_apc_conf_t` *const apc_conf, const `BOOL` is_host)
Configure APC .
- `vtss_rc vtss_phy_10g_apc_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `BOOL` is_host, `vtss_phy_10g_apc_conf_t` *const apc_conf)
Get configuration of APC .
- `vtss_rc vtss_phy_10g_apc_status_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `BOOL` is_host, `vtss_phy_10g_apc_status_t` *const apc_status)
Get status of APC.
- `vtss_rc vtss_phy_10g_apc_restart` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `BOOL` is_host)
Restart of APC - Debug function only.
- `vtss_rc vtss_phy_10g_jitter_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_phy_10g_jitter_conf_t` *const jitter_conf, `BOOL` is_host)
Configure optimised jitter.
- `vtss_rc vtss_phy_10g_jitter_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_10g_jitter_conf_t` *jitter_conf, `BOOL` is_host)
Gets current Jitter configuration.
- `vtss_rc vtss_phy_10g_jitter_status_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_10g_jitter_conf_t` *const jitter_conf, `BOOL` is_host)
Jitter status.
- `vtss_rc vtss_phy_10g_sync_clkout_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `BOOL` *const sync_clkout)
Get the status of recovered clock from PHY. (recommended to use vtss_phy_10g_rxckout_get instead)

- `vtss_rc vtss_phy_10g_sync_clkout_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `BOOL` sync_clkout)

Enable or Disable the recovered clock from PHY. (recommended to use `vtss_phy_10g_rxckout_set` instead)
- `vtss_rc vtss_phy_10g_xfp_clkout_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `BOOL` *const xfp_clkout)

Get the status of RXCLKOUT/TXCLKOUT from PHY. (recommended to use `vtss_phy_10g_txckout_get` instead)
- `vtss_rc vtss_phy_10g_xfp_clkout_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `BOOL` xfp_clkout)

Enable or Disable the RXCLKOUT/TXCLKOUT from PHY. (recommended to use `vtss_phy_10g_txckout_set` instead)
- `vtss_rc vtss_phy_10g_rxckout_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_10g_rxckout_conf_t` *const rxckout)

Get the rx recovered clock output configuration.
- `vtss_rc vtss_phy_10g_rxckout_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_phy_10g_rxckout_conf_t` *const rxckout)

Set the rx recovered clock output configuration.
- `vtss_rc vtss_phy_10g_txckout_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_10g_txckout_conf_t` *const txckout)

Get the status of tx recovered clock output configuration.
- `vtss_rc vtss_phy_10g_txckout_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_phy_10g_txckout_conf_t` *const txckout)

Set the tx recovered clock output configuration.
- `vtss_rc vtss_phy_10g_srefclk_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_10g_srefclk_mode_t` *const srefclk)

*Get the configuration of srefclk setting
Available for PHY family VENICE
This function should not be used any more, instead use the API function `vtss_phy_10g_mode_get`, see the parameter documentation for that function.*
- `vtss_rc vtss_phy_10g_srefclk_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_phy_10g_srefclk_mode_t` *const srefclk)

*Set the configuration of srefclk setting. Available for PHY family VENICE
This function should not be used any more, instead use the API function `vtss_phy_10g_mode_set`, see the parameter documentation for that function.*
- `vtss_rc vtss_phy_10g_sckout_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_phy_10g_sckout_conf_t` *const sckout)

Set the configuration of sckout setting. Available for PHY family MALIBU
- `vtss_rc vtss_phy_10g_ckout_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_phy_10g_ckout_conf_t` *const ckout)

Set the configuration of ckout setting. Available for PHY family MALIBU
- `vtss_rc vtss_phy_10g_line_clk_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_phy_10g_line_clk_conf_t` *const line_clk)

Set the configuration of sckout setting. Available for PHY family MALIBU
- `vtss_rc vtss_phy_10g_host_clk_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_phy_10g_host_clk_conf_t` *const host_clk)

Set the configuration of sckout setting. Available for PHY family MALIBU
- `vtss_rc vtss_phy_10g_line_recvrd_clk_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_phy_10g_line_clk_conf_t` *const line_clk)

Set the configuration of line clk recovered setting. Available for PHY family MALIBU
- `vtss_rc vtss_phy_10g_host_recvrd_clk_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_phy_10g_host_clk_conf_t` *const host_clk)

Set the configuration of line clk recovered setting. Available for PHY family MALIBU

Set the configuration of host clk recovered setting. Available for PHY family MALIBU

- `vtss_rc vtss_phy_10g_lane_sync_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_10g_lane_sync_conf_t *const lane_sync)`

Set the configuration of lane sync setting. Available for PHY family MALIBU

- `vtss_rc vtss_phy_10g_debug_register_dump (const vtss_inst_t inst, const vtss_debug_printf_t pr, BOOL clear, const vtss_port_no_t port_no)`

Set the configuration of 10G PHY Dump setting. Available for PHY family Venice & Malibu

- `vtss_rc vtss_phy_10g_base_kr_train_aneg_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_10g_base_kr_train_aneg_t *const kr_tr_aneg)`

Get the configuration of 10g_base_kr_tr_aneg setting. Available for PHY family Malibu,venice-c.

- `vtss_rc vtss_phy_10g_base_kr_train_aneg_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_10g_base_kr_train_aneg_t *const kr_tr_aneg)`

Set the configuration of 10g_base_kr_tr_aneg setting. Available for PHY family Malibu,venice-c.

- `vtss_rc vtss_phy_10g_base_host_kr_train_aneg_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_10g_base_kr_train_aneg_t *const kr_tr_aneg)`

Set the configuration of Host 10g_base_kr_tr_aneg setting. Available for PHY family Malibu.

- `vtss_rc vtss_phy_10g_base_kr_conf_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_10g_base_kr_conf_t *const kr_conf)`

Get the configuration of Line 10G output buffer setting. Available for PHY family VENICE-c,Malibu.

- `vtss_rc vtss_phy_10g_base_kr_conf_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_10g_base_kr_conf_t *const kr_conf)`

Set the configuration of Line 10G output buffer. Available for PHY family VENICE-c,MALIBU: Note: The parameters cm1,c0, c1 have a range of -32..31. The Output signal from the KR circuit is symmetric, i.e. the voltage output is configured value + 1/2lsb. (ex: -1 => -1/2lsb voltage level, 0 => +1/2lsb voltage level) The parameter ampl is set in steps of 25 mV, therefore the setting is rounded up to a multiplum og 25 mV, i.e. 1101..1125 => 1125 mVppd.

- `vtss_rc vtss_phy_10g_base_kr_host_conf_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_10g_base_kr_conf_t *const kr_conf)`

Get the configuration of Host 10G output buffer.

- `vtss_rc vtss_phy_10g_base_kr_host_conf_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_10g_base_kr_conf_t *const kr_conf)`

Set the configuration of host 10G output buffer. Note: The parameters cm1,c0, c1 have a range of -32..31. The Output signal from the KR circuit is symmetric, i.e. the voltage output is configured value + 1/2lsb. (ex: -1 => -1/2lsb voltage level, 0 => +1/2lsb voltage level) The parameter ampl is set in steps of 25 mV, therefore the setting is rounded up to a multiplum og 25 mV, i.e. 1101..1125 => 1125 mVppd.

- `vtss_rc vtss_phy_10g_kr_status_get (const vtss_inst_t inst, const vtss_port_no_t port_no, BOOL direction, vtss_phy_10g_base_kr_status_t *const kr_status)`

Get status of KR autonegotiation and training. Available for PHY family Malibu,Venice-c.

- `vtss_rc vtss_phy_10g_ob_status_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_10g_ob_status_t *const ob_status)`

Get the status of Output Buffer.

- `vtss_rc vtss_phy_10g_status_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_10g_status_t *const status)`

Get the link and fault status of the PHY sublayers.

- `vtss_rc vtss_phy_10g_serd़es_status_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_10g_serd़es_status_t *const status)`

Get the status of PHY including sub layers.

- `vtss_rc vtss_phy_10g_reset (const vtss_inst_t inst, const vtss_port_no_t port_no)`

Reset the phy. Phy is reset to default values.

- `vtss_rc vtss_phy_10g_clause_37_status_get (const vtss_inst_t inst, vtss_port_no_t port_no, vtss_phy_10g_clause_37_cmn_status_t *const status)`

Get clause 37 status.

- `vtss_rc vtss_phy_10g_clause_37_control_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_10g_clause_37_control_t` *const control)

Get clause 37 control configuration from software.
- `vtss_rc vtss_phy_10g_clause_37_control_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_phy_10g_clause_37_control_t` *const control)

Set clause 37 control configuration.
- `vtss_rc vtss_phy_10g_loopback_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_phy_10g_loopback_t` *const loopback)

Enable/Disable a phy network or system loopback.
Only one loopback mode can be active at the same time.
- `vtss_rc vtss_phy_10g_loopback_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_10g_loopback_t` *const loopback)

Get loopback settings.
- `vtss_rc vtss_phy_10g_cnt_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_10g_cnt_t` *const cnt)

Get counters.
- `vtss_rc vtss_phy_10g_power_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_10g_power_t` *const power)

Get the power settings.
- `vtss_rc vtss_phy_10g_power_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_phy_10g_power_t` *const power)

Set the power settings.
- `BOOL vtss_phy_10G_is_valid` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no)

Gives a True/False value if the Phy is supported by the API
Only Vitesse phys are supported. `vtss_phy_10g_mode_set()` must be applied.
- `vtss_rc vtss_phy_10g_failover_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_10g_failover_mode_t` *const mode)

Set the failover mode.
- `vtss_rc vtss_phy_10g_failover_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_10g_failover_mode_t` *const mode)

Get the failover mode.
- `vtss_rc vtss_phy_10g_auto_failover_set` (const `vtss_inst_t` inst, `vtss_phy_10g_auto_failover_conf_t` *const mode)

Set the automatic failover mode.
- `vtss_rc vtss_phy_10g_auto_failover_get` (const `vtss_inst_t` inst, `vtss_phy_10g_auto_failover_conf_t` *const mode)

Get the Automatic failover mode Configuration.
- `vtss_rc vtss_phy_10g_vscope_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_phy_10g_vscope_conf_t` *const conf)

set VSCOPE fast scan configuration
- `vtss_rc vtss_phy_10g_vscope_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_10g_vscope_conf_t` *const conf)

get VSCOPE fast scan configuration
- `vtss_rc vtss_phy_10g_vscope_scan_status_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_10g_vscope_scan_status_t` *const conf)

set VSCOPE fast scan configuration
- `vtss_rc vtss_phy_10g_pcs_prbs_gen_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_10g_pcs_prbs_gen_conf_t` *const conf, const `BOOL` line)

Set PCS-prbs generator Configuration.
- `vtss_rc vtss_phy_10g_pcs_prbs_gen_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_10g_pcs_prbs_gen_conf_t` *const conf, const `BOOL` line)

Get PCS-prbs generator Configuration.

- `vtss_rc vtss_phy_10g_pcs_prbs_mon_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_10g_pcs_prbs_mon_conf_t` *const conf, const `BOOL` line)

Set PCS-prbs monitor Configuration.
- `vtss_rc vtss_phy_10g_pcs_prbs_mon_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_10g_pcs_prbs_mon_conf_t` *const conf, const `BOOL` line)

Get PCS-prbs monitor Configuration.
- `vtss_rc vtss_phy_10g_pcs_prbs_mon_status_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_10g_pcs_prbs_mon_conf_t` *const conf, const `BOOL` line)

Get PCS-prbs monitor status.
- `vtss_rc vtss_phy_10g_prbs_gen_conf` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_10g_prbs_gen_conf_t` *const conf)

set prbs generator Configuration
- `vtss_rc vtss_phy_10g_prbs_gen_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_10g_prbs_gen_conf_t` *const conf, `BOOL` line)

get prbs generator Configuration
- `vtss_rc vtss_phy_10g_prbs_mon_conf` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_10g_prbs_mon_conf_t` *const conf)

set prbs generator Configuration
- `vtss_rc vtss_phy_10g_prbs_mon_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_10g_prbs_mon_conf_t` *const conf, `BOOL` line)

prbs generator Configuration get
- `vtss_rc vtss_phy_10g_prbs_mon_status_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_10g_prbs_mon_conf_t` *const mon_status, `BOOL` line, `BOOL` reset)

prbs Checker Status get
- `vtss_rc vtss_phy_10g_pkt_gen_conf` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_10g_pkt_gen_conf_t` *const conf)

Set Packet generation Configuration.
- `vtss_rc vtss_phy_10g_pkt_mon_conf` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `BOOL` ts_rd, `vtss_phy_10g_pkt_mon_conf_t` *const conf, `vtss_phy_10g_timestamp_val_t` *const conf_ts)

Set Packet Monitor Configuration.
- `vtss_rc vtss_phy_10g_pkt_mon_counters_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_10g_pkt_mon_conf_t` *const conf)

Set/Get Packet mon Counters.
- `vtss_rc vtss_phy_10g_id_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_10g_id_t` *const phy_id)

Read the Phy Id.
- `vtss_rc vtss_phy_10g_gpio_mode_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_gpio_10g_no_t` gpio_no, const `vtss_gpio_10g_gpio_mode_t` *const mode)

Set GPIO mode. There is only one set of GPIO per PHY chip - not per port.
- `vtss_rc vtss_phy_10g_gpio_mode_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_gpio_10g_no_t` gpio_no, `vtss_gpio_10g_gpio_mode_t` *const mode)

Get GPIO mode.
- `vtss_rc vtss_phy_10g_gpio_read` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_gpio_10g_no_t` gpio_no, `BOOL` *const value)

Read from GPIO input pin.
- `vtss_rc vtss_phy_10g_gpio_write` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_gpio_10g_no_t` gpio_no, const `BOOL` value)

Write to GPIO output pin.
- `vtss_rc vtss_phy_10g_event_enable_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_phy_10g_event_t` ev_mask, const `BOOL` enable)

Enabling / Disabling of events.
- `vtss_rc vtss_phy_10g_event_enable_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_10g_event_t` *const ev_mask)

- Get Enabling of events.*
- `vtss_rc vtss_phy_10g_extended_event_enable_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_10g_extnd_event_t` *const ex_ev_mask)
- Get Enabling of events.*
- `vtss_rc vtss_phy_10g_event_poll` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_10g_event_t` *const ev_mask)
- Polling for active events.*
- `vtss_rc vtss_phy_10g_pcs_status_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_10g_extnd_event_t` *const ex_events)
- poll and clear PCS STICKY Register*
- `vtss_rc vtss_phy_10g_extended_event_poll` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_10g_extnd_event_t` *const ex_events)
- Polling for active events.*
- `vtss_rc vtss_phy_10g_extended_event_enable_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_phy_10g_extnd_event_t` ex_ev_mask, const `BOOL` extnd_enable)
- Enabling / Disabling of events.*
- `vtss_rc vtss_phy_10g_poll_1sec` (const `vtss_inst_t` inst)
- Function is called once a second.*
- `vtss_rc vtss_phy_10g_edc_fw_status_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_10g_fw_status_t` *const status)
- Internal microprocessor status.*
- `vtss_rc vtss_phy_10g_fc_buffer_reset` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no)
- debug function for PHY 10G FC buffer reset*
- `vtss_rc vtss_phy_10g_csr_read` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `u32` dev, const `u32` addr, `u32` *const value)
- CSR register read.*
- `vtss_rc vtss_phy_10g_csr_write` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `u32` dev, const `u32` addr, const `u32` value)
- CSR register write.*
- `vtss_rc vtss_phy_warm_start_10g_failed_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no)
- Function for checking if any issue were seen during warm-start.*
- `vtss_rc vtss_phy_10g_sgmii_mode_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `BOOL` enable)
- Enables Pass through mode in 10G PHY.*
- `vtss_rc vtss_phy_10g_i2c_read` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `u16` addr, `u16` *value)
- read from i2c device*
- `vtss_rc vtss_phy_10g_i2c_write` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `u16` addr, const `u16` *value)
- Write to i2c device.*
- `vtss_rc vtss_phy_10g_get_user_data` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, void **user_data)
- Gets generic pointer in vtss_state structure.*

8.30.1 Detailed Description

10G PHY API

This header file describes 10G PHY control functions

8.30.2 Macro Definition Documentation

8.30.2.1 VTSS_SLEWRATE_25PS

```
#define VTSS_SLEWRATE_25PS 0
```

Slew rate ctrl of OB.

Slewrate = 25ps

Definition at line 1186 of file vtss_phy_10g_api.h.

8.30.2.2 VTSS_SLEWRATE_35PS

```
#define VTSS_SLEWRATE_35PS 1
```

Slewrate = 35ps

Definition at line 1187 of file vtss_phy_10g_api.h.

8.30.2.3 VTSS_SLEWRATE_55PS

```
#define VTSS_SLEWRATE_55PS 2
```

Slewrate = 55ps

Definition at line 1188 of file vtss_phy_10g_api.h.

8.30.2.4 VTSS_SLEWRATE_70PS

```
#define VTSS_SLEWRATE_70PS 3
```

Slewrate = 70ps

Definition at line 1189 of file vtss_phy_10g_api.h.

8.30.2.5 VTSS_SLEWRATE_120PS

```
#define VTSS_SLEWRATE_120PS 4
```

Slewrate = 120ps

Definition at line 1190 of file vtss_phy_10g_api.h.

8.30.2.6 VTSS_SLEWRATE_INVALID

```
#define VTSS_SLEWRATE_INVALID 5
```

Slewrate is invalid

Definition at line 1191 of file vtss_phy_10g_api.h.

8.30.2.7 BOOLEAN_STORAGE_COUNT

```
#define BOOLEAN_STORAGE_COUNT 6
```

BOOL parameters to be stored during Vscope Scan

Definition at line 1894 of file vtss_phy_10g_api.h.

8.30.2.8 UNSIGNED_STORAGE_COUNT

```
#define UNSIGNED_STORAGE_COUNT 5
```

UNSIGNED parameters to be stored during Vscope Scan

Definition at line 1895 of file vtss_phy_10g_api.h.

8.30.2.9 PHASE_POINTS

```
#define PHASE_POINTS 128
```

phase points range from 0-127

Definition at line 1915 of file vtss_phy_10g_api.h.

8.30.2.10 AMPLITUDE_POINTS

```
#define AMPLITUDE_POINTS 64
```

amplitude points range from 0-63

Definition at line 1916 of file vtss_phy_10g_api.h.

8.30.2.11 VTSS_PHY_10G_ONE_LINE_ACTIVE

```
#define VTSS_PHY_10G_ONE_LINE_ACTIVE 0x08
```

Bit indicating PHY with only one line interface

Definition at line 2261 of file vtss_phy_10g_api.h.

8.30.2.12 VTSS_PHY_10G_MACSEC_DISABLED

```
#define VTSS_PHY_10G_MACSEC_DISABLED 0x04
```

Bit indicating that macsec is disabled

Definition at line 2262 of file vtss_phy_10g_api.h.

8.30.2.13 VTSS_PHY_10G_TIMESTAMP_DISABLED

```
#define VTSS_PHY_10G_TIMESTAMP_DISABLED 0x02
```

Bit indicating that timestamp feature is disabled

Definition at line 2263 of file vtss_phy_10g_api.h.

8.30.2.14 VTSS_PHY_10G_MACSEC_KEY_128

```
#define VTSS_PHY_10G_MACSEC_KEY_128 0x01
```

Bit indicating that only 128 bit macsec encryption key is supported, otherwise it is 128/256 key

Definition at line 2264 of file vtss_phy_10g_api.h.

8.30.2.15 VTSS_10G_PHY_GPIO_MAX

```
#define VTSS_10G_PHY_GPIO_MAX 12
```

Max value of gpio_no parameter

Definition at line 2496 of file vtss_phy_10g_api.h.

8.30.2.16 VTSS_10G_PHY_GPIO_MAL_MAX

```
#define VTSS_10G_PHY_GPIO_MAL_MAX 40
```

Max value of gpio_no parameter, Malibu

Definition at line 2498 of file vtss_phy_10g_api.h.

8.30.2.17 VTSS_PHY_10G_LINK_LOS_EV

```
#define VTSS_PHY_10G_LINK_LOS_EV 0x00000001
```

Event source identification mask values.

PHY Link Los interrupt - only on 8486

Definition at line 2563 of file vtss_phy_10g_api.h.

8.30.2.18 VTSS_PHY_10G_RX_LOL_EV [1/2]

```
#define VTSS_PHY_10G_RX_LOL_EV 0x00000002
```

PHY RXLOL interrupt - only on 8488

PHY RX LOL interrupt - 8256 specific

Definition at line 2603 of file vtss_phy_10g_api.h.

8.30.2.19 VTSS_PHY_10G_TX_LOL_EV [1/2]

```
#define VTSS_PHY_10G_TX_LOL_EV 0x00000004
```

PHY TXLOL interrupt - only on 8488

PHY TX LOL interrupt - 8256 specific

Definition at line 2604 of file vtss_phy_10g_api.h.

8.30.2.20 VTSS_PHY_10G_LOPC_EV

```
#define VTSS_PHY_10G_LOPC_EV 0x00000008
```

PHY LOPC interrupt - only on 8488

Definition at line 2566 of file vtss_phy_10g_api.h.

8.30.2.21 VTSS_PHY_10G_HIGH_BER_EV

```
#define VTSS_PHY_10G_HIGH_BER_EV 0x00000010
```

PHY HIGH_BER interrupt - only on 8488

Definition at line 2567 of file vtss_phy_10g_api.h.

8.30.2.22 VTSS_PHY_10G_MODULE_STAT_EV

```
#define VTSS_PHY_10G_MODULE_STAT_EV 0x00000020
```

PHY MODULE_STAT interrupt - only on 8488

Definition at line 2568 of file vtss_phy_10g_api.h.

8.30.2.23 VTSS_PHY_10G_PCS_RECEIVE_FAULT_EV

```
#define VTSS_PHY_10G_PCS_RECEIVE_FAULT_EV 0x00000040
```

PHY PCS_RECEIVE_FAULT interrupt - only on 8488

Definition at line 2569 of file vtss_phy_10g_api.h.

8.30.2.24 VTSS_PHY_EWIS_SEF_EV

```
#define VTSS_PHY_EWIS_SEF_EV 0x00000080
```

SEF has changed state - only for 8488

Definition at line 2571 of file vtss_phy_10g_api.h.

8.30.2.25 VTSS_PHY_EWIS_FPLM_EV

```
#define VTSS_PHY_EWIS_FPLM_EV 0x00000100
```

far-end (PLM-P) / (LCDP) - only for 8488

Definition at line 2572 of file vtss_phy_10g_api.h.

8.30.2.26 VTSS_PHY_EWIS_FAIS_EV

```
#define VTSS_PHY_EWIS_FAIS_EV 0x00000200
```

far-end (AIS-P) / (LOP) - only for 8488

Definition at line 2573 of file vtss_phy_10g_api.h.

8.30.2.27 VTSS_PHY_EWIS_LOF_EV

```
#define VTSS_PHY_EWIS_LOF_EV 0x00000400
```

Loss of Frame (LOF) - only for 8488

Definition at line 2574 of file vtss_phy_10g_api.h.

8.30.2.28 VTSS_PHY_EWIS_RDIL_EV

```
#define VTSS_PHY_EWIS_RDIL_EV 0x00000800
```

Line Remote Defect Indication (RDI-L) - only for 8488

Definition at line 2575 of file vtss_phy_10g_api.h.

8.30.2.29 VTSS_PHY_EWIS_AISL_EV

```
#define VTSS_PHY_EWIS_AISL_EV 0x00001000
```

Line Alarm Indication Signal (AIS-L) - only for 8488

Definition at line 2576 of file vtss_phy_10g_api.h.

8.30.2.30 VTSS_PHY_EWIS_LCDP_EV

```
#define VTSS_PHY_EWIS_LCDP_EV 0x00002000
```

Loss of Code-group Delineation (LCD-P) - only for 8488

Definition at line 2577 of file vtss_phy_10g_api.h.

8.30.2.31 VTSS_PHY_EWIS_PLMP_EV

```
#define VTSS_PHY_EWIS_PLMP_EV 0x00004000
```

Path Label Mismatch (PLMP) - only for 8488

Definition at line 2578 of file vtss_phy_10g_api.h.

8.30.2.32 VTSS_PHY_EWIS_AISP_EV

```
#define VTSS_PHY_EWIS_AISP_EV 0x00008000
```

Path Alarm Indication Signal (AIS-P) - only for 8488

Definition at line 2579 of file vtss_phy_10g_api.h.

8.30.2.33 VTSS_PHY_EWIS_LOPP_EV

```
#define VTSS_PHY_EWIS_LOPP_EV 0x00010000
```

Path Loss of Pointer (LOP-P) - only for 8488

Definition at line 2580 of file vtss_phy_10g_api.h.

8.30.2.34 VTSS_PHY_EWIS_UNEQP_EV

```
#define VTSS_PHY_EWIS_UNEQP_EV 0x00020000
```

Unequiped Path (UNEQ-P) - only for 8488

Definition at line 2581 of file vtss_phy_10g_api.h.

8.30.2.35 VTSS_PHY_EWIS_FEUNEQP_EV

```
#define VTSS_PHY_EWIS_FEUNEQP_EV 0x00040000
```

Far-end Unequiped Path (UNEQ-P) - only for 8488

Definition at line 2582 of file vtss_phy_10g_api.h.

8.30.2.36 VTSS_PHY_EWIS_FERDIP_EV

```
#define VTSS_PHY_EWIS_FERDIP_EV 0x00080000
```

Far-end Path Remote Defect Identifier (RDI-P) - only for 8488

Definition at line 2583 of file vtss_phy_10g_api.h.

8.30.2.37 VTSS_PHY_EWIS_REIL_EV

```
#define VTSS_PHY_EWIS_REIL_EV 0x00100000
```

Line Remote Error Indication (REI-L) - only for 8488

Definition at line 2584 of file vtss_phy_10g_api.h.

8.30.2.38 VTSS_PHY_EWIS_REIP_EV

```
#define VTSS_PHY_EWIS_REIP_EV 0x00200000
```

Path Remote Error Indication (REI-P) - only for 8488

Definition at line 2585 of file vtss_phy_10g_api.h.

8.30.2.39 VTSS_PHY_EWIS_B1_NZ_EV

```
#define VTSS_PHY_EWIS_B1_NZ_EV 0x00400000
```

PMTICK B1 BIP (B1_ERR_CNT) not zero - only for 8488

Definition at line 2586 of file vtss_phy_10g_api.h.

8.30.2.40 VTSS_PHY_EWIS_B2_NZ_EV

```
#define VTSS_PHY_EWIS_B2_NZ_EV 0x00800000
```

PMTICK B2 BIP (B1_ERR_CNT) not zero - only for 8488

Definition at line 2587 of file vtss_phy_10g_api.h.

8.30.2.41 VTSS_PHY_EWIS_B3_NZ_EV

```
#define VTSS_PHY_EWIS_B3_NZ_EV 0x01000000
```

PMTICK B3 BIP (B1_ERR_CNT) not zero - only for 8488

Definition at line 2588 of file vtss_phy_10g_api.h.

8.30.2.42 VTSS_PHY_EWIS_REIL_NZ_EV

```
#define VTSS_PHY_EWIS_REIL_NZ_EV 0x02000000
```

PMTICK REI-L (REIL_ERR_CNT) not zero - only for 8488

Definition at line 2589 of file vtss_phy_10g_api.h.

8.30.2.43 VTSS_PHY_EWIS_REIP_NZ_EV

```
#define VTSS_PHY_EWIS_REIP_NZ_EV 0x04000000
```

PMTICK REI-P (REIP_ERR_CNT) not zero - only for 8488

Definition at line 2590 of file vtss_phy_10g_api.h.

8.30.2.44 VTSS_PHY_EWIS_B1_THRESH_EV

```
#define VTSS_PHY_EWIS_B1_THRESH_EV 0x08000000
```

B1_THRESH_ERR - only for 8488

Definition at line 2591 of file vtss_phy_10g_api.h.

8.30.2.45 VTSS_PHY_EWIS_B2_THRESH_EV

```
#define VTSS_PHY_EWIS_B2_THRESH_EV 0x10000000
```

B2_THRESH_ERR - only for 8488

Definition at line 2592 of file vtss_phy_10g_api.h.

8.30.2.46 VTSS_PHY_EWIS_B3_THRESH_EV

```
#define VTSS_PHY_EWIS_B3_THRESH_EV 0x20000000
```

B3_THRESH_ERR - only for 8488

Definition at line 2593 of file vtss_phy_10g_api.h.

8.30.2.47 VTSS_PHY_EWIS_REIL_THRESH_EV

```
#define VTSS_PHY_EWIS_REIL_THRESH_EV 0x40000000
```

REIL_THRESH_ERR - only for 8488

Definition at line 2594 of file vtss_phy_10g_api.h.

8.30.2.48 VTSS_PHY_EWIS_REIP_THRESH_EV

```
#define VTSS_PHY_EWIS_REIP_THRESH_EV 0x80000000
```

REIP_THRESH_ERR - only for 8488

Definition at line 2595 of file vtss_phy_10g_api.h.

8.30.2.49 VTSS_PHY_10G_RX_LOS_EV

```
#define VTSS_PHY_10G_RX_LOS_EV 0x00000001
```

PHY RX LOS interrupt - 8256 specific

Definition at line 2602 of file vtss_phy_10g_api.h.

8.30.2.50 VTSS_PHY_10G_RX_LOL_EV [2/2]

```
#define VTSS_PHY_10G_RX_LOL_EV 0x00000002
```

PHY RX LOL interrupt - only on 8488

PHY RX LOL interrupt - 8256 specific

Definition at line 2603 of file vtss_phy_10g_api.h.

8.30.2.51 VTSS_PHY_10G_TX_LOL_EV [2/2]

```
#define VTSS_PHY_10G_TX_LOL_EV 0x00000004
```

PHY TXLOL interrupt - only on 8488

PHY TX LOL interrupt - 8256 specific

Definition at line 2604 of file vtss_phy_10g_api.h.

8.30.2.52 VTSS_PHY_10G_RX_CHAR_DEC_CNT_THRESH_EV

```
#define VTSS_PHY_10G_RX_CHAR_DEC_CNT_THRESH_EV 0x00000010
```

PHY RX character decode error - 8256 specific

Definition at line 2606 of file vtss_phy_10g_api.h.

8.30.2.53 VTSS_PHY_10G_TX_CHAR_ENC_CNT_THRESH_EV

```
#define VTSS_PHY_10G_TX_CHAR_ENC_CNT_THRESH_EV 0x00000020
```

PHY TX character encode error count - 8256 specific

Definition at line 2607 of file vtss_phy_10g_api.h.

8.30.2.54 VTSS_PHY_10G_RX_BLK_DEC_CNT_THRESH_EV

```
#define VTSS_PHY_10G_RX_BLK_DEC_CNT_THRESH_EV 0x00000040
```

PHY RX block decode error count - 8256 specific

Definition at line 2608 of file vtss_phy_10g_api.h.

8.30.2.55 VTSS_PHY_10G_TX_BLK_ENC_CNT_THRESH_EV

```
#define VTSS_PHY_10G_TX_BLK_ENC_CNT_THRESH_EV 0x00000080
```

PHY TX block encode error count- 8256 specific

Definition at line 2609 of file vtss_phy_10g_api.h.

8.30.2.56 VTSS_PHY_10G_RX_SEQ_CNT_THRESH_EV

```
#define VTSS_PHY_10G_RX_SEQ_CNT_THRESH_EV 0x00000100
```

PHY RX sequencing error count - 8256 specific

Definition at line 2610 of file vtss_phy_10g_api.h.

8.30.2.57 VTSS_PHY_10G_TX_SEQ_CNT_THRESH_EV

```
#define VTSS_PHY_10G_TX_SEQ_CNT_THRESH_EV 0x00000200
```

PHY TX sequencing error count - 8256 specific

Definition at line 2611 of file vtss_phy_10g_api.h.

8.30.2.58 VTSS_PHY_10G_FEC_UNFIXED_CNT_THRESH_EV

```
#define VTSS_PHY_10G_FEC_UNFIXED_CNT_THRESH_EV 0x00000400
```

PHY KR-FEC uncorrectable block count interrupt - 8256 specific

Definition at line 2612 of file vtss_phy_10g_api.h.

8.30.2.59 VTSS_PHY_10G_FEC_FIXED_CNT_THRESH_EV

```
#define VTSS_PHY_10G_FEC_FIXED_CNT_THRESH_EV 0x00000800
```

PHY KR-FEC corrected threshold - 8256 specific

Definition at line 2613 of file vtss_phy_10g_api.h.

8.30.2.60 VTSS_PHY_10G_HIGBTER_EV

```
#define VTSS_PHY_10G_HIGBTER_EV 0x00001000
```

PHY high bit Error - 8256 specific

Definition at line 2614 of file vtss_phy_10g_api.h.

8.30.2.61 VTSS_PHY_10G_RX_LINK_STAT_EV [1/2]

```
#define VTSS_PHY_10G_RX_LINK_STAT_EV 0x00002000
```

PHY Link status up/down interrupt - 8256 specific

Definition at line 2616 of file vtss_phy_10g_api.h.

8.30.2.62 VTSS_PHY_10G_RX_LINK_STAT_EV [2/2]

```
#define VTSS_PHY_10G_RX_LINK_STAT_EV 0x00002000
```

PHY Link status up/down interrupt - 8256 specific

Definition at line 2616 of file vtss_phy_10g_api.h.

8.30.2.63 VTSS_PHY_10G_GPIO_INT_AGG0_EV

```
#define VTSS_PHY_10G_GPIO_INT_AGG0_EV 0x00004000
```

PHY GPIO interrupt on Aggregator0 - 8256 specific

Definition at line 2617 of file vtss_phy_10g_api.h.

8.30.2.64 VTSS_PHY_10G_GPIO_INT_AGG1_EV

```
#define VTSS_PHY_10G_GPIO_INT_AGG1_EV 0x00008000
```

PHY GPIO interrupt on Aggregator1 - 8256 specific

Definition at line 2618 of file vtss_phy_10g_api.h.

8.30.2.65 VTSS_PHY_10G_GPIO_INT_AGG2_EV

```
#define VTSS_PHY_10G_GPIO_INT_AGG2_EV 0x00010000
```

PHY GPIO interrupt on Aggregator2 - 8256 specific

Definition at line 2619 of file vtss_phy_10g_api.h.

8.30.2.66 VTSS_PHY_10G_GPIO_INT_AGG3_EV

```
#define VTSS_PHY_10G_GPIO_INT_AGG3_EV 0x00020000
```

PHY GPIO interrupt on Aggregator3 - 8256 specific

Definition at line 2620 of file vtss_phy_10g_api.h.

8.30.2.67 VTSS_PHY_1G_LINE_AUTONEG_RESTART_EV

```
#define VTSS_PHY_1G_LINE_AUTONEG_RESTART_EV 0x00040000
```

PHY 1G Line side Autoneg restart event

Definition at line 2621 of file vtss_phy_10g_api.h.

8.30.2.68 VTSS_PHY_1G_HOST_AUTONEG_RESTART_EV

```
#define VTSS_PHY_1G_HOST_AUTONEG_RESTART_EV 0x00080000
```

PHY 1G Host side Autoneg restart event - 8256 specific

Definition at line 2622 of file vtss_phy_10g_api.h.

8.30.2.69 VTSS_PHY_10G_LINE_MAC_LOCAL_FAULT_EV

```
#define VTSS_PHY_10G_LINE_MAC_LOCAL_FAULT_EV 0x00100000
```

PHY 10G LINE MAC local fault event

Definition at line 2625 of file vtss_phy_10g_api.h.

8.30.2.70 VTSS_PHY_10G_HOST_MAC_LOCAL_FAULT_EV

```
#define VTSS_PHY_10G_HOST_MAC_LOCAL_FAULT_EV 0x00400000
```

PHY 10G HOST MAC local fault event

Definition at line 2626 of file vtss_phy_10g_api.h.

8.30.2.71 VTSS_PHY_10G_LINE_MAC_REMOTE_FAULT_EV

```
#define VTSS_PHY_10G_LINE_MAC_REMOTE_FAULT_EV 0x00800000
```

PHY 10G LINE MAC remote fault event

Definition at line 2627 of file vtss_phy_10g_api.h.

8.30.2.72 VTSS_PHY_10G_HOST_MAC_REMOTE_FAULT_EV

```
#define VTSS_PHY_10G_HOST_MAC_REMOTE_FAULT_EV 0x01000000
```

PHY 10G HOST MAC remote fault event

Definition at line 2628 of file vtss_phy_10g_api.h.

8.30.3 Typedef Documentation

8.30.3.1 vtss_gpio_10g_no_t

```
typedef u16 vtss_gpio_10g_no_t
```

GPIO type for 10G ports

Definition at line 412 of file vtss_phy_10g_api.h.

8.30.3.2 ckout_sel_t

```
typedef enum ckout_sel_ ckout_sel_t
```

10G Phy CKOUTs Enum

Malibu Only

8.30.3.3 vtss_32_cntr_t

```
typedef u32 vtss_32_cntr_t
```

32-bit counter

Definition at line 2175 of file vtss_phy_10g_api.h.

8.30.3.4 vtss_phy_10g_event_t

```
typedef u32 vtss_phy_10g_event_t
```

The type definition to contain the above defined event mask

Definition at line 2597 of file vtss_phy_10g_api.h.

8.30.3.5 vtss_phy_10g_extnd_event_t

```
typedef u32 vtss_phy_10g_extnd_event_t
```

The type definition to contain the above defined extended event mask

Definition at line 2631 of file vtss_phy_10g_api.h.

8.30.4 Enumeration Type Documentation

8.30.4.1 oper_mode_t

```
enum oper_mode_t
```

10G Phy operating mode enum type

Enumerator

VTSS_PHY_LAN_MODE	LAN mode: Single clock (XREFCK=156,25 MHz), no recovered clock output
VTSS_PHY_WAN_MODE	WAN mode: 848X: Dual clock (XREFCK=156,25 MHz, WREFCK=155,52 MHz), no recovered clock output Venice: Single clock (XREFCK), no recovered clock output
VTSS_PHY_1G_MODE	8488: 1G pass-through mode Venice: 1G mode, Single clock (XREFCK=156,25 MHz), no recovered clock output For 1G operation, customer should select VTSS_MEDIA_TYPE_SR for all media applications and specify the operation is in 1G mode
VTSS_PHY_LAN_SYNCE_MODE	LAN SyncE: if hl_clk_synth == 1: 8488: Single clock (XREFCK=156,25 MHz), recovered clock output enabled Venice: Single clock (XREFCK=156,25 MHz), recovered clock output enabled if hl_clk_synth == 0: 8488: Dual clock (XREFCK=156,25 MHz, SREFCK=156,25 MHz), recovered clock output enabled Venice: Dual clock (XREFCK=156,25 MHz, SREFCK=156,25 MHz), recovered clock output enabled

Enumerator

VTSS_PHY_WAN_SYNCE_MODE	<p>WAN SyncE: if hl_clk_synth == 1: 8488: Single clock (WREFCK=155,52 MHz or 622,08 MHz), recovered clock output enabled Venice: Single clock (XREFCK=156,25 MHz), recovered clock output enabled if hl_clk_synth == 0: 8488: Dual clock (WREFCK=155,52 MHz or 622,08 MHz, SREFCK=155,52 MHz), recovered clock output enabled Venice: Dual clock (XREFCK=156,25 MHz, SREFCK=155,52 MHz), recovered clock output enabled</p>
VTSS_PHY_LAN_MIXED_SYNCE_MODE	8488: Channels are in different modes, channel being configured is in LAN Venice: Same as VTSS_PHY_LAN_SYNCE_MODE
VTSS_PHY_WAN_MIXED_SYNCE_MODE	8488: Channels are in different modes, channel being configured is in WAN Venice: Same as VTSS_PHY_WAN_SYNCE_MODE
VTSS_PHY_REPEATERT_MODE	Malibu: Repeater mode,better jitter performance

Definition at line 45 of file vtss_phy_10g_api.h.

8.30.4.2 vtss_wrefclk_t

enum [vtss_wrefclk_t](#)

Modes for WAN reference clock.

Enumerator

VTSS_WREFCLK_155_52	WREFCLK = 155.52Mhz - WAN ref clock
VTSS_WREFCLK_622_08	WREFCLK = 622.08Mhz - WAN ref clock

Definition at line 80 of file vtss_phy_10g_api.h.

8.30.4.3 vtss_phy_interface_mode

enum [vtss_phy_interface_mode](#)

Phy Interface modes.

Enumerator

VTSS_PHY_XAUI_XFI	XAUI <-> XFI - Interface mode.
VTSS_PHY_XGMII_XFI	XGMII <-> XFI - Interface mode. Only for VSC8486

Enumerator

<code>VTSS_PHY_RXAUI_XFI</code>	RXAUI <-> XFI - Interface mode. Only for Venice
<code>VTSS_PHY_SGMII_LANE_0_XFI</code>	SGMII <-> XFI - LANE 0. Only for Venice
<code>VTSS_PHY_SGMII_LANE_3_XFI</code>	SGMII <-> XFI - LANE 3. Only for Venice
<code>VTSS_PHY_SFI_XFI</code>	SFI <-> XFI - Interface mode. Only for Malibu

Definition at line 94 of file vtss_phy_10g_api.h.

8.30.4.4 vtss_recvrd_t

enum `vtss_recvrd_t`

Modes for recovered clock.

Enumerator

<code>VTSS_RECVRD_RXCLKOUT</code>	RXCLKOUT is used for recovered clock
<code>VTSS_RECVRD_TXCLKOUT</code>	TXCLKOUT is used for recovered clock

Definition at line 104 of file vtss_phy_10g_api.h.

8.30.4.5 vtss_recvrdclk_cdr_div_t

enum `vtss_recvrdclk_cdr_div_t`

Modes for recovered clock divisor.

Enumerator

<code>VTSS_RECVRDCLK_CDR_DIV_64</code>	recovered clock is /64
<code>VTSS_RECVRDCLK_CDR_DIV_66</code>	recovered clock is /66

Definition at line 110 of file vtss_phy_10g_api.h.

8.30.4.6 vtss_srefclk_div_t

enum `vtss_srefclk_div_t`

Modes for Synch-E recovered clock.

Enumerator

VTSS_SREFCLK_DIV_64	SREFCLK/64 ,valid for LAN,WAN
VTSS_SREFCLK_DIV_66	SREFCLK/66 ,valid for LAN
VTSS_SREFCLK_DIV_16	SREFCLK/16 ,valid for WAN

Definition at line 116 of file vtss_phy_10g_api.h.

8.30.4.7 vtss_wref_clk_div_t

```
enum vtss_wref_clk_div_t
```

Modes for WREFCLK clock divisor.

Enumerator

VTSS_WREFCLK_NONE	NA
VTSS_WREFCLK_DIV_16	WREFCLK/16

Definition at line 124 of file vtss_phy_10g_api.h.

8.30.4.8 apc_ib_regulator_t

```
enum apc_ib_regulator_t
```

APC Rx regulator mode.

Enumerator

VTSS_AP_C_IB_SFP_PLUS_ZR	SFP+ ZR module.
VTSS_AP_C_IB_BACKPLANE	Backplane application.

Definition at line 130 of file vtss_phy_10g_api.h.

8.30.4.9 ddr_mode_t

```
enum ddr_mode_t
```

Interleave mode.

Enumerator

VTSS_DDR_MODE_A	Interleave mode with A alignment symbol based byte re-ordering
VTSS_DDR_MODE_K	Interleave mode with K coma based byte re-ordering
VTSS_DDR_MODE_M	Interleave mode with A alignment and 8b10b decoding disabled

Definition at line 136 of file vtss_phy_10g_api.h.

8.30.4.10 clk_mstr_t

```
enum clk\_mstr\_t
```

Clock master.

Enumerator

VTSS_CLK_MSTR_INTERNAL	Master clock is internal
VTSS_CLK_MSTR_EXTERNAL	Master clock is external

Definition at line 143 of file vtss_phy_10g_api.h.

8.30.4.11 vtss_rptr_rate_t

```
enum vtss\_rptr\_rate\_t
```

Repeater Data rate.

Enumerator

VTSS_RPTR_RATE_NONE	None
VTSS_RPTR_RATE_10_3125	LAN rate=10.3125 Gbps,
VTSS_RPTR_RATE_9_9532	WAN rate=9.9532 Gbps
VTSS_RPTR_RATE_11_3	rate=11.3 Gbps,clock 171Mhz
VTSS_RPTR_RATE_10_5187	Fiber channel rate=10.51875 Gbps,
VTSS_RPTR_RATE_1_25	1G rate=1.25Gbps
VTSS_RPTR_RATE_10_709	OTU2 rate= 10.709 Gbps
VTSS_RPTR_RATE_11_095727	OTU2E rate = 11.095727 Gbps
VTSS_RPTR_RATE_11_05	OTU1E rate = 11.05 Gbps

Definition at line 149 of file vtss_phy_10g_api.h.

8.30.4.12 vtss_phy_10g_media_t

enum [vtss_phy_10g_media_t](#)

10G Phy Media type

Enumerator

VTSS_MEDIA_TYPE_SR	SR,10GBASE-SR
VTSS_MEDIA_TYPE_SR2	SR,10GBASE-SR
VTSS_MEDIA_TYPE_DAC	DAC,Direct attach cable
VTSS_MEDIA_TYPE_ZR	ZR,10GBASE-ZR
VTSS_MEDIA_TYPE_KR	KR,10GBASE-KR
VTSS_MEDIA_TYPE_SR_SC	SR,10GBASE-SR with smart control
VTSS_MEDIA_TYPE_SR2_SC	SR,10GBASE-SR with smart control
VTSS_MEDIA_TYPE_DAC_SC	DAC,Direct attach cable with smart control
VTSS_MEDIA_TYPE_ZR_SC	ZR,10GBASE-ZR with smart control
VTSS_MEDIA_TYPE_ZR2_SC	ZR,10GBASE-ZR with smart control with ld_lev_ini:40
VTSS_MEDIA_TYPE_KR_SC	KR,10GBASE-KR with smart control
VTSS_MEDIA_TYPE_NONE	None

Definition at line 170 of file vtss_phy_10g_api.h.

8.30.4.13 vtss_phy_6g_link_partner_distance_t

enum [vtss_phy_6g_link_partner_distance_t](#)

6G serdes link partner distance selection

Enumerator

VTSS_6G_LINK_SHORT_RANGE	distance between 6G macro and serdes macro of link partner is less (direct connection)
VTSS_6G_LINK_LONG_RANGE	distance between 6G macro and serdes macro of link parter is more (connection via backplanes)

Definition at line 186 of file vtss_phy_10g_api.h.

8.30.4.14 vtss_phy_10g_ib_apc_op_mode_t

enum [vtss_phy_10g_ib_apc_op_mode_t](#)

10G SERDES APC operation

Enumerator

VTSS_IB_APP_AUTO	AUTO Operation
VTSS_IB_APP_MANUAL	Manual operation
VTSS_IB_APP_FREEZE	Freeze
VTSS_IB_APP_RESET	Reset
VTSS_IB_APP_RESTART	Restart APC
VTSS_IB_APP_NONE	None

Definition at line 197 of file vtss_phy_10g_api.h.

8.30.4.15 vtss_channel_t

```
enum vtss_channel_t
```

Channel modes - Auto is recommended.

Enumerator

VTSS_CHANNEL_AUTO	Automatically detects the channel id based on the phy order. The phys be setup in the consecutive order, from the lowest MDIO to highest MDIO address
VTSS_CHANNEL_0	Channel id is hardcoded to 0
VTSS_CHANNEL_1	Channel id is hardcoded to 1
VTSS_CHANNEL_2	Channel id is hardcoded to 2
VTSS_CHANNEL_3	Channel id is hardcoded to 3

Definition at line 321 of file vtss_phy_10g_api.h.

8.30.4.16 vtss_recvrd_clkout_t

```
enum vtss_recvrd_clkout_t
```

Modes for (rx/tx) recovered clock output.

Enumerator

VTSS_RECVRD_CLKOUT_DISABLE	recovered clock output is disabled
VTSS_RECVRD_CLKOUT_LINE_SIDE_RX_CLOCK	recovered clock output is derived from Lineside Rx clock
VTSS_RECVRD_CLKOUT_LINE_SIDE_TX_CLOCK	recovered clock output is derived from Lineside Tx clock

Definition at line 699 of file vtss_phy_10g_api.h.

8.30.4.17 vtss_phy_10g_srefclk_freq_t

enum [vtss_phy_10g_srefclk_freq_t](#)

10G Phy sref clock input frequency

Enumerator

VTSS_PHY_10G_SREFCLK_156_25	156,25 MHz
VTSS_PHY_10G_SREFCLK_125_00	125,00 MHz
VTSS_PHY_10G_SREFCLK_155_52	155,52 MHz
VTSS_PHY_10G_SREFCLK_INVALID	Other values are not allowed

Definition at line 779 of file vtss_phy_10g_api.h.

8.30.4.18 vtss_phy_10g_ckout_freq_t

enum [vtss_phy_10g_ckout_freq_t](#)

10G Phy clock frequency

Malibu only

Enumerator

VTSS_PHY_10G_CLK_FULL_RATE	LAN:332.25 MHz, WAN:311.04MHz, 1G:125MHz
VTSS_PHY_10G_CLK_DIVIDE_BY_2	LAN:161.12 MHz, WAN:155.52MHz, 1G:62.5MHz
VTSS_PHY_10G_CLK_INVALID	Other values are not allowed

Definition at line 831 of file vtss_phy_10g_api.h.

8.30.4.19 vtss_ckout_data_sel_t

enum [vtss_ckout_data_sel_t](#)

Modes for recovered clock output.

Applicable to Malibu only

Enumerator

VTSS_CKOUT_LINE0_TX_CLOCK	Line0 Transmit clock
VTSS_CKOUT_LINE1_TX_CLOCK	Line1 Transmit clock
VTSS_CKOUT_LINE2_TX_CLOCK	Line2 Transmit clock
VTSS_CKOUT_LINE3_TX_CLOCK	Line3 Transmit clock

Enumerator

VTSS_CKOUT_HOST0_TX_CLOCK	Host0 Transmit clock
VTSS_CKOUT_HOST1_TX_CLOCK	Host1 Transmit clock
VTSS_CKOUT_HOST2_TX_CLOCK	Host2 Transmit clock
VTSS_CKOUT_HOST3_TX_CLOCK	Host3 Transmit clock
VTSS_CKOUT_LINE0_RECVRD_CLOCK	Line0 Recovered clock
VTSS_CKOUT_LINE1_RECVRD_CLOCK	Line1 Recovered clock
VTSS_CKOUT_LINE2_RECVRD_CLOCK	Line2 Recovered clock
VTSS_CKOUT_LINE3_RECVRD_CLOCK	Line3 Recovered clock
VTSS_CKOUT_HOST0_RECVRD_CLOCK	Host0 Recovered clock
VTSS_CKOUT_HOST1_RECVRD_CLOCK	Host1 Recovered clock
VTSS_CKOUT_HOST2_RECVRD_CLOCK	Host2 Recovered clock
VTSS_CKOUT_HOST3_RECVRD_CLOCK	Host3 Recovered clock
VTSS_CKOUT_HOST_PLL_CLOCK	Host PLL clock
VTSS_CKOUT_LINE_PLL_CLOCK	Line PLL clock
VTSS_CKOUT_CSR_CLOCK	CSR clock
VTSS_CKOUT_LTC_CLOCK	LTC clock
VTSS_CKOUT_DF2F_CLOCK	Df2f clock
VTSS_CKOUT_F2DF_CLOCK	F2df clock
VTSS_CKOUT_DEBUG1	Debug1
VTSS_CKOUT_DEBUG2	Debug2
VTSS_CKOUT_OSCILLATOR_OUTPUT	Oscillator output

Definition at line 841 of file vtss_phy_10g_api.h.

8.30.4.20 vtss_phy_10g_squelch_src_t

enum [vtss_phy_10g_squelch_src_t](#)

squelch control source

Applicable to Malibu only

Enumerator

VTSS_CKOUT_SQUELCH_SRC_GPIO0	GPIO0 as source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_GPIO1	GPIO1 as source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_GPIO2	GPIO2 as source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_GPIO3	GPIO3 as source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_GPIO4	GPIO4 as source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_GPIO5	GPIO5 as source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_GPIO6	GPIO6 as source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_GPIO7	GPIO7 as source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_LINK_LINE0	Link status from Line0 source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_LINK_LINE1	Link status from Line1 source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_LINK_LINE2	Link status from Line2 source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_LINK_LINE3	Link status from Line3 source of auto squelch

Enumerator

VTSS_CKOUT_SQUELCH_SRC_LINK_HOST0	Link status from Host0 source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_LINK_HOST1	Link status from Host1 source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_LINK_HOST2	Link status from Host2 source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_LINK_HOST3	Link status from Host3 source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_LOS_LINE0	Serdes LOS from Line0 as source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_LOS_LINE1	Serdes LOS from Line1 as source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_LOS_LINE2	Serdes LOS from Line2 as source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_LOS_LINE3	Serdes LOS from Line3 as source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_LOS_HOST0	Serdes LOS from Host0 as source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_LOS_HOST1	Serdes LOS from Host1 as source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_LOS_HOST2	Serdes LOS from Host2 as source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_LOS_HOST3	Serdes LOS from Host3 as source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_LINK_LINE0_KR	Link status from Line0 KR source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_LINK_LINE1_KR	Link status from Line1 KR source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_LINK_LINE2_KR	Link status from Line2 KR source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_LINK_LINE3_KR	Link status from Line3 KR source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_LINK_HOST0_KR	Link status from Host0 KR source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_LINK_HOST1_KR	Link status from Host1 KR source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_LINK_HOST2_KR	Link status from Host2 KR source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_LINK_HOST3_KR	Link status from Host3 KR source of auto squelch
VTSS_CKOUT_NO_SQUELCH	No squelch(32-63)

Definition at line 873 of file vtss_phy_10g_api.h.

8.30.4.21 vtss_phy_10g_clk_sel_t

```
enum vtss_phy_10g_clk_sel_t
```

Modes of recovered clocks for ckout and sckout pins.

Applicable to Malibu only

Enumerator

VTSS_PHY_10G_LINE0_RECVRD_CLOCK	Line0 Recovered clock
VTSS_PHY_10G_LINE1_RECVRD_CLOCK	Line1 Recovered clock
VTSS_PHY_10G_LINE2_RECVRD_CLOCK	Line2 Recovered clock
VTSS_PHY_10G_LINE3_RECVRD_CLOCK	Line3 Recovered clock
VTSS_PHY_10G_HOST0_RECVRD_CLOCK	Host0 Recovered clock
VTSS_PHY_10G_HOST1_RECVRD_CLOCK	Host1 Recovered clock
VTSS_PHY_10G_HOST2_RECVRD_CLOCK	Host2 Recovered clock
VTSS_PHY_10G_HOST3_RECVRD_CLOCK	Host3 Recovered clock
VTSS_PHY_10G_SREFCLK	SREFCLK
VTSS_PHY_10G_SYNC_DISABLE	Sync Disable 9-15

Definition at line 914 of file vtss_phy_10g_api.h.

8.30.4.22 vtss_phy_10g_recvrd_clk_sel_t

enum [vtss_phy_10g_recvrd_clk_sel_t](#)

Modes of recovered clock selection.

Applicable to Malibu only

Enumerator

VTSS_PHY_10G_USE_LINE0_RECVRD_CLOCK	All lines using Line0 Recovered clock
VTSS_PHY_10G_USE_LINE1_RECVRD_CLOCK	All lines using Line1 Recovered clock
VTSS_PHY_10G_USE_LINE2_RECVRD_CLOCK	All lines using Line2 Recovered clock
VTSS_PHY_10G_USE_LINE3_RECVRD_CLOCK	All lines using Line3 Recovered clock
VTSS_PHY_10G_USE_HOST0_RECVRD_CLOCK	All lines using Host0 Recovered clock
VTSS_PHY_10G_USE_HOST1_RECVRD_CLOCK	All lines using Host1 Recovered clock
VTSS_PHY_10G_USE_HOST2_RECVRD_CLOCK	All lines using Host2 Recovered clock
VTSS_PHY_10G_USE_HOST3_RECVRD_CLOCK	All lines using Host3 Recovered clock
VTSS_PHY_10G_USE_SREFCLK_CLOCK	All lines using SREFCLK
VTSS_PHY_10G_USE_DEFAULT_RECVRD_CLK↔OCK	Use Recvrd Clk from the respctive line LineX Uses recovered clock from LineX only

Definition at line 932 of file vtss_phy_10g_api.h.

8.30.4.23 ckout_sel_

enum [ckout_sel_](#)

10G Phy CKOUTs Enum

Malibu Only

Definition at line 951 of file vtss_phy_10g_api.h.

8.30.4.24 vtss_phy_10g_sckout_freq_t

enum [vtss_phy_10g_sckout_freq_t](#)

10G Phy sckout clock input frequency

Enumerator

VTSS_PHY_10G_SCKOUT_156_25	156,25 MHz
VTSS_PHY_10G_SCKOUT_125_00	125,00 MHz
VTSS_PHY_10G_SCKOUT_INVALID	Other values are not allowed

Definition at line 972 of file vtss_phy_10g_api.h.

8.30.4.25 vtss_phy_10g_rx_macro_t

enum [vtss_phy_10g_rx_macro_t](#)

10G Phy Rx MACRO Configuration

Malibu Only

Enumerator

VTSS_PHY_10G_RX_MACRO_LINE	Rx MACRO Line
VTSS_PHY_10G_RX_MACRO_HOST	Rx MACRO Host
VTSS_PHY_10G_RX_MACRO_SREFCLK	Rx MACRO SREFCLK

Definition at line 1110 of file vtss_phy_10g_api.h.

8.30.4.26 vtss_phy_10g_tx_macro_t

enum [vtss_phy_10g_tx_macro_t](#)

10G Phy tx MACRO Configuration

Malibu Only

Enumerator

VTSS_PHY_10G_TX_MACRO_LINE	Tx MACRO Line
VTSS_PHY_10G_TX_MACRO_HOST	Tx MACRO Host
VTSS_PHY_10G_TX_MACRO_SCKOUT	Tx MACRO SREFCLK

Definition at line 1120 of file vtss_phy_10g_api.h.

8.30.4.27 vtss_phy_10g_clause_37_remote_fault_t

enum [vtss_phy_10g_clause_37_remote_fault_t](#)

Auto-negotiation remote fault type.

Advertisement Word (Refer to IEEE 802.3 Clause 37): MSB LSB D15 D14 D13 D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0 +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+ | NP | Ack| RF2| R↔ F1|rsvd|rsvd|rsvd| PS2| PS1| HD | FD |rsvd|rsvd|rsvd|rsvd|rsvd| +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

Enumerator

VTSS_PHY_10G_CLAUSE_37_RF_LINK_OK	Link OK
VTSS_PHY_10G_CLAUSE_37_RF_OFFLINE	Off line
VTSS_PHY_10G_CLAUSE_37_RF_LINK_FAILURE	Link failure
VTSS_PHY_10G_CLAUSE_37_RF_AUTONEG_ERROR	Autoneg error

Definition at line 1497 of file vtss_phy_10g_api.h.

8.30.4.28 vtss_lb_type_t

enum [vtss_lb_type_t](#)

10G loopback types

Enumerator

VTSS_LB_NONE	No looback
VTSS_LB_SYSTEM_XS_SHALLOW	System Loopback B, XAUI -> XS -> XAUI 4x800E.13, Venice: H2
VTSS_LB_SYSTEM_XS_DEEP	System Loopback C, XAUI -> XS -> XAUI 4x800F.2, Venice: N.A.
VTSS_LB_SYSTEM_PCS_SHALLOW	System Loopback E, XAUI -> PCS FIFO -> XAUI 3x8005.2, Venice: N.A.
VTSS_LB_SYSTEM_PCS_DEEP	System Loopback G, XAUI -> PCS -> XAUI 3x0000.14, Venice: H3
VTSS_LB_SYSTEM_PMA	System Loopback J, XAUI -> PMA -> XAUI 1x0000.0, Venice: H4
VTSS_LB_NETWORK_XS_SHALLOW	Network Loopback D, XFI -> XS -> XFI 4x800F.1, Venice: N.A.
VTSS_LB_NETWORK_XS_DEEP	Network Loopback A, XFI -> XS -> XFI 4x0000.1 4x800E.13=0, Venice: L1
VTSS_LB_NETWORK_PCS	Network Loopback F, XFI -> PCS -> XFI 3x8005.3, Venice: L2
VTSS_LB_NETWORK_WIS	Network Loopback H, XFI -> WIS -> XFI 2xE600.0, Venice: N.A.
VTSS_LB_NETWORK_PMA	Network Loopback K, XFI -> PMA -> XFI 1x8000.8, Venice: L3
VTSS_LB_H2	Host Loopback 2, 40-bit XAUI-PHY interface Mirror XAUI data
VTSS_LB_H3	Host Loopback 3, 64-bit PCS after the gearbox FF00 repeating IEEE PCS system loopback
VTSS_LB_H4	Host Loopback 4, 64-bit WIS FF00 repeating IEEE WIS system loopback
VTSS_LB_H5	Host Loopback 5, 1-bit SFP+ after SerDes Mirror XAUI data IEEE PMA system loopback
VTSS_LB_H6	Host Loopback 6, 32-bit XAUI-PHY interface Mirror XAUI data
VTSS_LB_L0	Line Loopback 0, 4-bit XAUI before SerDes Mirror SFP+ data
VTSS_LB_L1	Line Loopback 1, 4-bit XAUI after SerDes Mirror SFP+ data IEEE PHY-XS network loopback
VTSS_LB_L2	Line Loopback 2, 64-bit XGMII after FIFO Mirror SFP+ data
VTSS_LB_L3	Line Loopback 3, 64-bit PMA interface Mirror SFP+ data

Definition at line 1598 of file vtss_phy_10g_api.h.

8.30.4.29 vtss_phy_10g_power_t

enum `vtss_phy_10g_power_t`

10G Phy power setting

Enumerator

<code>VTSS_PHY_10G_POWER_ENABLE</code>	Enable Phy power for all sublayers
<code>VTSS_PHY_10G_POWER_DISABLE</code>	Disable Phy power for all sublayers

Definition at line 1699 of file vtss_phy_10g_api.h.

8.30.4.30 vtss_phy_10g_failover_mode_t

enum `vtss_phy_10g_failover_mode_t`

10G Phy Failover Mode Setting

Enumerator

<code>VTSS_PHY_10G_PMA_TO_FROM_XAUI_NORMAL</code>	PMA_0/1 to XAUI_0/1. 8487: XAUI 0 to PMA 0
<code>VTSS_PHY_10G_PMA_TO_FROM_XAUI_CROSSED_SED</code>	PMA_0/1 to XAUI_1/0. 8487: XAUI 1 to PMA 0
<code>VTSS_PHY_10G_PMA_0_TO_FROM_XAUI_0_TO_XAUI_1</code>	PMA 0 to/from XAUI 0 and to XAUI 1
<code>VTSS_PHY_10G_PMA_0_TO_FROM_XAUI_1_TO_XAUI_0</code>	PMA 0 to/from XAUI 1 and to XAUI 0
<code>VTSS_PHY_10G_PMA_1_TO_FROM_XAUI_0_TO_XAUI_1</code>	PMA 1 to/from XAUI 0 and to XAUI 1. VSC8487:N/A
<code>VTSS_PHY_10G_PMA_1_TO_FROM_XAUI_1_TO_XAUI_0</code>	PMA 1 to/from XAUI 1 and to XAUI 0. VSC8487:N/A

Definition at line 1749 of file vtss_phy_10g_api.h.

8.30.4.31 vtss_phy_10g_auto_failover_event_t

enum `vtss_phy_10g_auto_failover_event_t`

10G Phy Automatic Failover Event Setting

Enumerator

VTSS_PHY_10G_AUTO_FAILOVER_EVENT_PCS_LINK_STATUS	PCS link status
VTSS_PHY_10G_AUTO_FAILOVER_EVENT_SERDES_LOS	LOS from SERDES
VTSS_PHY_10G_AUTO_FAILOVER_EVENT_WIS_LOF	LOF from Line WIS
VTSS_PHY_10G_AUTO_FAILOVER_EVENT_GPIO	External GPIO input
VTSS_PHY_10G_AUTO_FAILOVER_EVENT_NONE	Manual switching to be done

Definition at line 1788 of file vtss_phy_10g_api.h.

8.30.4.32 vtss_phy_10g_auto_failover_filter_t

enum [vtss_phy_10g_auto_failover_filter_t](#)

10G PHY Automatic Failover Filter

Enumerator

VTSS_PHY_10G_AUTO_FAILOVER_FILTER_NONE	No filter configuration
VTSS_PHY_10G_AUTO_FAILOVER_FILTER_CNT_B2316	False condition, upper 8 bits of 24-bit threshold
VTSS_PHY_10G_AUTO_FAILOVER_FILTER_CNT_B70	False condition, lower 8 bits of 24-bit threshold
VTSS_PHY_10G_AUTO_FAILOVER_FILTER_CNT_A2316	True condition, upper 8 bits of 24-bit threshold
VTSS_PHY_10G_AUTO_FAILOVER_FILTER_CNT_A70	True condition, lower 8 bits of 24-bit threshold

Definition at line 1798 of file vtss_phy_10g_api.h.

8.30.4.33 vtss_phy_10g_vscope_scan_t

enum [vtss_phy_10g_vscope_scan_t](#)

VSCOPE scan types.

Enumerator

VTSS_PHY_10G_FAST_SCAN	selects the fast scan feature
VTSS_PHY_10G_FAST_SCAN_PLUS	selects the fast scan feature with diagonal points
VTSS_PHY_10G_QUICK_SCAN	selects the quick scan feature
VTSS_PHY_10G_FULL_SCAN	selects the full scan feature

Definition at line 1848 of file vtss_phy_10g_api.h.

8.30.4.34 vtss_phy_10g_pkt_mon_rst_t

enum [vtss_phy_10g_pkt_mon_rst_t](#)

10G PHY Packet monitor configuration

Enumerator

VTSS_PHY_10G_PKT_MON_RST_ALL	Reset all counters
VTSS_PHY_10G_PKT_MON_RST_GOOD	Reset good crc counter
VTSS_PHY_10G_PKT_MON_RST_BAD	Reset bad crc counter
VTSS_PHY_10G_PKT_MON_RST_FRAG	Reset Fragment counter
VTSS_PHY_10G_PKT_MON_RST_LFAULT	Reset local fault counter
VTSS_PHY_10G_PKT_MON_RST_BER	Reset Ber counter
VTSS_PHY_10G_PKT_MON_RST_NONE	None

Definition at line 2165 of file vtss_phy_10g_api.h.

8.30.4.35 vtss_10g_phy_gpio_t

enum [vtss_10g_phy_gpio_t](#)

GPIO configured mode.

GPIO configured mode

Enumerator

VTSS_10G_PHY_GPIO_NOT_INITIALIZED	This GPIO pin has been initialized by a call to API from application. registers contain power-up default value
VTSS_10G_PHY_GPIO_OUT	Output enabled
VTSS_10G_PHY_GPIO_IN	Input enabled
VTSS_10G_PHY_GPIO_WIS_INT	Output WIS interrupt channel 0 or 1 (depending on port_no) enabled
VTSS_10G_PHY_GPIO_1588_LOAD_SAVE	Input interrupt generated on falling edge Input interrupt generated on raising edge Input interrupt generated on raising and falling edge Input 1588 load/save function
VTSS_10G_PHY_GPIO_1588_1PPS_0	Output 1588 1PPS from channel 0 function
VTSS_10G_PHY_GPIO_1588_1PPS_1	Output 1588 1PPS from channel 1 function
VTSS_10G_PHY_GPIO_1588_1PPS_2	Output 1588 1PPS from channel 2 function
VTSS_10G_PHY_GPIO_1588_1PPS_3	Output 1588 1PPS from channel 3 function
VTSS_10G_PHY_GPIO_PCS_RX_FAULT	PCS_RX_FAULT (from channel 0 or 1) is transmitted on GPIO
VTSS_10G_PHY_GPIO_SET_I2C_MASTER	Used in communicating with I2C slave, like SPP+
VTSS_10G_PHY_GPIO_TX_ENABLE	Transmit enable , MALIBU
VTSS_10G_PHY_GPIO_LINE_PLL_STATUS	Line PLL Status , MALIBU
VTSS_10G_PHY_GPIO_HOST_PLL_STATUS	Host PLL Status , MALIBU
VTSS_10G_PHY_GPIO_RCOMP_BUSY	RCOMP busy Status , MALIBU
VTSS_10G_PHY_GPIO_CHAN_INT_0	Interrupt 0 from channel , MALIBU

Enumerator

VTSS_10G_PHY_GPIO_CHAN_INT_1	Interrupt 1 from channel , MALIBU
VTSS_10G_PHY_GPIO_1588_INT	1588 Interrupt , MALIBU
VTSS_10G_PHY_GPIO_TS_FIFO_EMPTY	TS FIFO empty , MALIBU
VTSS_10G_PHY_GPIO_AGG_INT_0	Aggregated interrupt 0 , MALIBU
VTSS_10G_PHY_GPIO_AGG_INT_1	Aggregated interrupt 1 , MALIBU
VTSS_10G_PHY_GPIO_AGG_INT_2	Aggregated interrupt 2 , MALIBU
VTSS_10G_PHY_GPIO_AGG_INT_3	Aggregated interrupt 3 , MALIBU
VTSS_10G_PHY_GPIO_PLL_INT_0	Interrupt 0 from PLL , MALIBU
VTSS_10G_PHY_GPIO_PLL_INT_1	Interrupt 0 from PLL , MALIBU
VTSS_10G_PHY_GPIO_SET_I2C_SLAVE	I2C Slave set , MALIBU
VTSS_10G_PHY_GPIO_CRSS_INT	Cross Connect Interrupt , MALIBU
VTSS_10G_PHY_GPIO_LED	LED Setting , MALIBU
VTSS_10G_PHY_GPIO_DRIVE_LOW	GPIO output to LOW , MALIBU
VTSS_10G_PHY_GPIO_DRIVE_HIGH	GPIO output to HIGH , MALIBU

Definition at line 2306 of file vtss_phy_10g_api.h.

8.30.4.36 `vtss_gpio_10g_gpio_intr_sgnl_t`

```
enum vtss\_gpio\_10g\_gpio\_intr\_sgnl\_t
```

GPIO internal signal types.

Enumerator

VTSS_10G_GPIO_INTR_SGNL_I2C_MSTR_CLK↔_OUT	GPIO output I2C master data out
VTSS_10G_GPIO_INTR_SGNL_LED_TX	GPIO output I2C master clock out
VTSS_10G_GPIO_INTR_SGNL_LED_RX	LED transmit
VTSS_10G_GPIO_INTR_SGNL_RX_ALARM	LED receive
VTSS_10G_GPIO_INTR_SGNL_TX_ALARM	RX Alarm
VTSS_10G_GPIO_INTR_SGNL_HOST_LINK	TX Alarm
VTSS_10G_GPIO_INTR_SGNL_LINE_LINK	Host Link status
VTSS_10G_GPIO_INTR_SGNL_LINE_KR_8b10b↔_2GPIO	Line Link status
VTSS_10G_GPIO_INTR_SGNL_LINE_KR_10b_2↔_GPIO	KR 8b10b
VTSS_10G_GPIO_INTR_SGNL_ROSI_PULSE	KR 10b
VTSS_10G_GPIO_INTR_SGNL_ROSI_SDATA	ROSI Pulse
VTSS_10G_GPIO_INTR_SGNL_ROSI_SCLK	ROSI sdata
VTSS_10G_GPIO_INTR_SGNL_TOSI_PULSE	ROSI sclock
VTSS_10G_GPIO_INTR_SGNL_TOSI_SCLK	TOSI Pulse
VTSS_10G_GPIO_INTR_SGNL_LINE_PCS1G_LINK	TOSI sclock
VTSS_10G_GPIO_INTR_SGNL_LINE_PCS_RX_↔_STAT	Line PCS1G link status

Enumerator

VTSS_10G_GPIO_INTR_SGNL_CLIENT_PCS1G_LINK	Line PCS RX link status
VTSS_10G_GPIO_INTR_SGNL_HOST_PCS_RX_STAT	Client PCS1G link status
VTSS_10G_GPIO_INTR_SGNL_HOST_SD10G_IB_SIG	Host PCS RX status
VTSS_10G_GPIO_INTR_SGNL_LINE_SD10G_IB_SIG	Host SERDES 10G 1B signal
VTSS_10G_GPIO_INTR_SGNL_HPCS_INTR	Line SERDES 10G 1B signal
VTSS_10G_GPIO_INTR_SGNL_LPCS_INTR	HPCS interrupt
VTSS_10G_GPIO_INTR_SGNL_CLIENT_PCS1G_INTR	LPCS interrupt
VTSS_10G_GPIO_INTR_SGNL_LINE_PCS1G_INTR	Client PCS1G interrupt
VTSS_10G_GPIO_INTR_SGNL_WIS_INT0	Line PCS1G interrupt
VTSS_10G_GPIO_INTR_SGNL_HOST_PMA_INT	WIS interrupt 0
VTSS_10G_GPIO_INTR_SGNL_LINE_PMA_INT	Host PMA interrupt
VTSS_10G_GPIO_INTR_SGNL_DATA_ACT_TX	Line PMA interrupt
VTSS_10G_GPIO_INTR_SGNL_DATA_ACT_RX	TX data activity
VTSS_10G_GPIO_INTR_SGNL_HDATA_ACT_TX	RX data activity
VTSS_10G_GPIO_INTR_SGNL_HDATA_ACT_RX	Host TX data activity
	Host RX data activity
VTSS_10G_GPIO_INTR_SGNL_XGMII_PAUS_EGR	
VTSS_10G_GPIO_INTR_SGNL_XGMII_PAUS_ING	XGMI pause egress
VTSS_10G_GPIO_INTR_SGNL_RX_PCS_PAUS	XGMI pause ingress
VTSS_10G_GPIO_INTR_SGNL_TX_PCS_PAUS	PCS RX Pause
VTSS_10G_GPIO_INTR_SGNL_RX_WIS_PAUS	PCS TX Pause
VTSS_10G_GPIO_INTR_SGNL_TX_WIS_PAUS	WIS RX Pause
VTSS_10G_GPIO_INTR_SGNL_ETH_CHAN_DIS	WIS TX Pause
VTSS_10G_GPIO_INTR_SGNL_MACSEC_1588_SFD_LANE	Ethernet channel disable
	MACSEC,1588 SFD lane
VTSS_10G_GPIO_INTR_SGNL_LINE_S_TX_FAULT	
VTSS_10G_GPIO_INTR_SGNL_LPCS1G_LATEN_CY0_OR_EWIS_BIT0	TX fault
VTSS_10G_GPIO_INTR_SGNL_LPCS1G_LATEN_CY1_OR_EWIS_BIT1	LPCS1G latency 0 in case of 1G mode or EWIS BIT 0
VTSS_10G_GPIO_INTR_SGNL_LPCS1G_CHAR_POS0_OR_EWIS_BIT2	LPCS1G latency 0 in case of 1G mode or EWIS BIT 1
VTSS_10G_GPIO_INTR_SGNL_LPCS1G_CHAR_POS1_OR_EWIS_WORD0	LPCS1G Char pos 0 in case of 1G mode or EWIS BIT 2
VTSS_10G_GPIO_INTR_SGNL_LPCS1G_CHAR_POS2_OR_EWIS_WORD1	LPCS1G Char pos 1 in case of 1G mode or EWIS word 0
VTSS_10G_GPIO_INTR_SGNL_LPCS1G_CHAR_POS3_OR_EWIS_WORD2	LPCS1G Char pos 2 in case of 1G mode or EWIS word 1
VTSS_10G_GPIO_INTR_SGNL_MACSEC_IGR_PRED_VAR0	LPCS1G Char pos 3 in case of 1G mode or EWIS word 2
VTSS_10G_GPIO_INTR_SGNL_MACSEC_IGR_PRED_VAR1	Macsec ingress predictor var 0
VTSS_10G_GPIO_INTR_SGNL_KR_ACTV_2GPIO	Macsec ingress predictor var 1
VTSS_10G_GPIO_INTR_SGNL_DFT_TX_2GPIO	KR activity

Enumerator

VTSS_10G_GPIO_INTR_SGNL_RESERVED	DFT transmit
VTSS_10G_GPIO_INTR_SGNL_EXE_LST_2GPIO_0	Reserved for future use
VTSS_10G_GPIO_INTR_SGNL_EXE_LST_2GPIO_1	EXE LST to GPIO 0
VTSS_10G_GPIO_INTR_SGNL_EXE_LST_2GPIO_2	EXE LST to GPIO 1
VTSS_10G_GPIO_INTR_SGNL_EXE_LST_2GPIO_3	EXE LST to GPIO 2
VTSS_10G_GPIO_INTR_SGNL_EXE_LST_2GPIO_4	EXE LST to GPIO 3
VTSS_10G_GPIO_INTR_SGNL_LINK_HCD_2GPIO_0	EXE LST to GPIO 4
VTSS_10G_GPIO_INTR_SGNL_LINK_HCD_2GPIO_1	Link HCD to GPIO 0
VTSS_10G_GPIO_INTR_SGNL_LINK_HCD_2GPIO_2	Link HCD to GPIO 1
VTSS_10G_GPIO_INTR_SGNL_ETH_1G_ENA	Link HCD to GPIO 2
VTSS_10G_GPIO_INTR_SGNL_H_KR_8b10b_2GPIO	Ethernet 1G enable
VTSS_10G_GPIO_INTR_SGNL_H_KR_10Gb_2GPIO	KR 8b10b to GPIO
VTSS_10G_GPIO_INTR_SGNL_H_KR_ACTV_2GPIO	KR10Gb to GPIO
VTSS_10G_GPIO_INTR_SGNL_NONE	KR activity to GPIO

Definition at line 2347 of file vtss_phy_10g_api.h.

8.30.4.37 vtss_gpio_10g_chan_intrpt_t

```
enum vtss_gpio_10g_chan_intrpt_t
```

GPIO Channel level interrupts.

Internal signals supported on Malibu

Enumerator

VTSS_10G_GPIO_INTRPT_WIS1	WIS interrupt 0
VTSS_10G_GPIO_INTRPT_LPCS10G	WIS interrupt 1
VTSS_10G_GPIO_INTRPT_HPCS10G	LPCS 10G interrupt
VTSS_10G_GPIO_INTRPT_LPCS1G	HPCS 10G interrupt
VTSS_10G_GPIO_INTRPT_HPCS1G	LPCS 1G interrupt
VTSS_10G_GPIO_INTRPT_MSEC_EGR	HPCS 1G interrupt
VTSS_10G_GPIO_INTRPT_MSEC_IGR	Macsec Egress interrupt
VTSS_10G_GPIO_INTRPT_LMAC	Macsec Ingress interrupt
VTSS_10G_GPIO_INTRPT_HMAC	Line MAC interrupt
VTSS_10G_GPIO_INTRPT_FCBUF	Host MAC interrupt

Enumerator

VTSS_10G_GPIO_INTRPT_LIGR_FIFO	FC Buffer interrupt
VTSS_10G_GPIO_INTRPT_LEGR_FIFO	Line ingress FIFO interrupt
VTSS_10G_GPIO_INTRPT_HEGR_FIFO	Line egress FIFO interrupt
VTSS_10G_GPIO_INTRPT_LPMA	Host egress FIFO interrupt
VTSS_10G_GPIO_INTRPT_HPMA	Line PMA interrupt

Definition at line 2419 of file vtss_phy_10g_api.h.

8.30.4.38 vtss_gpio_10g_aggr_intrpt_t

```
enum vtss_gpio_10g_aggr_intrpt_t
```

GPIO Channel level interrupts.

Channel Level Interrupts

Enumerator

VTSS_10G_GPIO_AGGR_INTRPT_CH0_INTR1_EN	CH0_INTR0_EN
VTSS_10G_GPIO_AGGR_INTRPT_CH1_INTR0_EN	CH0_INTR1_EN
VTSS_10G_GPIO_AGGR_INTRPT_CH1_INTR1_EN	CH1_INTR0_EN
VTSS_10G_GPIO_AGGR_INTRPT_CH2_INTR0_EN	CH1_INTR1_EN
VTSS_10G_GPIO_AGGR_INTRPT_CH2_INTR1_EN	CH2_INTR0_EN
VTSS_10G_GPIO_AGGR_INTRPT_CH3_INTR0_EN	CH2_INTR1_EN
VTSS_10G_GPIO_AGGR_INTRPT_CH3_INTR1_EN	CH3_INTR0_EN
VTSS_10G_GPIO_AGGR_INTRPT_IP1588_0_INTR0_EN	CH3_INTR1_EN
VTSS_10G_GPIO_AGGR_INTRPT_IP1588_0_INTR1_EN	IP1588_0_INTR0_EN
VTSS_10G_GPIO_AGGR_INTRPT_IP1588_0_INTR2_EN	IP1588_0_INTR1_EN
VTSS_10G_GPIO_AGGR_INTRPT_IP1588_0_INTR3_EN	IP1588_0_INTR2_EN
VTSS_10G_GPIO_AGGR_INTRPT_IP1588_1_INTR0_EN	IP1588_0_INTR3_EN
VTSS_10G_GPIO_AGGR_INTRPT_IP1588_1_INTR1_EN	TS_FIFO empty channel 0
VTSS_10G_GPIO_AGGR_INTRPT_IP1588_1_INTR2_EN	TS_FIFO empty channel 1
VTSS_10G_GPIO_AGGR_INTRPT_IP1588_1_INTR3_EN	TS_FIFO empty channel 2
VTSS_10G_GPIO_AGGR_INTRPT_LCPLL_0_INTR_EN	TS_FIFO empty channel 3
VTSS_10G_GPIO_AGGR_INTRPT_LCPLL_1_INTR_EN	LCPLL_0_INTR_EN
VTSS_10G_GPIO_AGGR_INTRPT_EXP4_INTR_EN	LCPLL_1_INTR_EN
VTSS_10G_GPIO_AGGR_INTRPT_CLK_MUX_INTR_EN	EXP4_INTR_EN
VTSS_10G_GPIO_AGGR_INTRPT_GPIO_INTR_EN	CLK_MUX_INTR_EN

Definition at line 2442 of file vtss_phy_10g_api.h.

8.30.4.39 vtss_gpio_10g_input_t

```
enum vtss_gpio_10g_input_t
```

GPIO Channel level interrupts.

Aggregated Interrupts

Enumerator

VTSS_10G_GPIO_INPUT_LINE_LOPC	Input that doesn't need any extra configuration
VTSS_10G_GPIO_INPUT_HOST_LOPC	LOPC from SFP on LINE

Definition at line 2470 of file vtss_phy_10g_api.h.

8.30.5 Function Documentation

8.30.5.1 vtss_phy_10g_mode_get()

```
vtss_rc vtss_phy_10g_mode_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_10g_mode_t *const mode )
```

Get the Phy operating mode.

Prior using this API, [vtss_phy_10g_mode_set\(\)](#) or [vtss_phy_10g_init\(\)](#) has to be called atleast once on this port/channel

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>mode</i>	[IN] Mode configuration.

Returns

VTSS_RC_OK on success.
VTSS_RC_ERROR on error.

8.30.5.2 vtss_phy_10g_init()

```
vtss_rc vtss_phy_10g_init (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_10g_init_parm_t *const init_conf )
```

Identify PHY and initialize software accordingly.

This API initializes the mode to 10G LAN. It supports AUTO and MANUAL channel Configuration. For AUTO channel Assignment the API should be called in the channel order (0 -> 3) For MANUAL channel Assignment the API can be called in any order, Application must provide the correct channel number specific to a port while calling Manual channel Assignment.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>init_conf</i>	[IN] Configuration.

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR on error.

8.30.5.3 vtss_phy_10g_mode_set()

```
vtss_rc vtss_phy_10g_mode_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_10g_mode_t *const mode )
```

Identify, Reset and set the operating mode of the PHY.

This API is supposed be called on base channel/port first and later for alternate channels/ports

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>mode</i>	[IN] Mode configuration.

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR on error.

8.30.5.4 vtss_phy_10g_ib_conf_set()

```
vtss_rc vtss_phy_10g_ib_conf_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_10g_ib_conf_t *const ib_conf,
    BOOL is_host )
```

Configure Input buffer .

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>ib_conf</i>	[IN] IB configuration.
<i>is_host</i>	[IN] Direction to be configured.

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR on error.

8.30.5.5 vtss_phy_10g_ib_conf_get()

```
vtss_rc vtss_phy_10g_ib_conf_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const BOOL is_host,
    vtss_phy_10g_ib_conf_t *const ib_conf )
```

Get configuration of Input buffer .

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>is_host</i>	[IN] Direction to be configured.
<i>ib_conf</i>	[OUT] IB configuration.

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR on error.

8.30.5.6 vtss_phy_10g_ib_status_get()

```
vtss_rc vtss_phy_10g_ib_status_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_10g_ib_status_t *const ib_status )
```

Get status of Input buffer .

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>ib_status</i>	[OUT] IB status.

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR on error.

8.30.5.7 vtss_phy_10g_apc_conf_set()

```
vtss_rc vtss_phy_10g_apc_conf_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_10g_apc_conf_t *const apc_conf,
    const BOOL is_host )
```

Configure APC .

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>apc_conf</i>	[IN] APC configuration.
<i>is_host</i>	[IN] Configuration side.

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR on error.

8.30.5.8 vtss_phy_10g_apc_conf_get()

```
vtss_rc vtss_phy_10g_apc_conf_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const BOOL is_host,
    vtss_phy_10g_apc_conf_t *const apc_conf )
```

Get configuration of APC .

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>is_host</i>	[IN] Configuration side.
<i>apc_conf</i>	[OUT] APC configuration.

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR on error.

8.30.5.9 vtss_phy_10g_apc_status_get()

```
vtss_rc vtss_phy_10g_apc_status_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const BOOL is_host,
    vtss_phy_10g_apc_status_t *const apc_status )
```

Get status of APC.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>is_host</i>	[IN] Configuration side.
<i>apc_status</i>	[OUT] APC status.

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR on error.

8.30.5.10 vtss_phy_10g_apc_restart()

```
vtss_rc vtss_phy_10g_apc_restart (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const BOOL is_host )
```

Restart of APC - Debug function only.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>is_host</i>	[IN] Configuration side.

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR on error.

8.30.5.11 vtss_phy_10g_jitter_conf_set()

```
vtss_rc vtss_phy_10g_jitter_conf_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_10g_jitter_conf_t *const jitter_conf,
    BOOL is_host )
```

Configure optimised jitter.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>jitter_conf</i>	[IN] Jitter configuration.
<i>is_host</i>	[IN] Direction to be configured.

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR on error.

8.30.5.12 vtss_phy_10g_jitter_conf_get()

```
vtss_rc vtss_phy_10g_jitter_conf_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_10g_jitter_conf_t * jitter_conf,
    BOOL is_host )
```

Gets current Jitter configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>jitter_conf</i>	[IN] Jitter configuration.
<i>is_host</i>	[IN] Direction to be configured.

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR on error.

8.30.5.13 vtss_phy_10g_jitter_status_get()

```
vtss_rc vtss_phy_10g_jitter_status_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_10g_jitter_conf_t *const jitter_conf,
    BOOL is_host )
```

Jitter status.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>jitter_conf</i>	[IN] Jitter configuration status.
<i>is_host</i>	[IN] Direction.

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR on error.

8.30.5.14 vtss_phy_10g_synce_clkout_get()

```
vtss_rc vtss_phy_10g_synce_clkout_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    BOOL *const synce_clkout )
```

Get the status of recovered clock from PHY. (recommended to use vtss_phy_10g_rxckout_get instead)

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>sync_clkout</i>	[IN] Recovered clock configuration.

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR on error.

8.30.5.15 vtss_phy_10g_sync_clkout_set()

```
vtss_rc vtss_phy_10g_sync_clkout_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const BOOL sync_clkout )
```

Enable or Disable the recovered clock from PHY. (recommended to use vtss_phy_10g_rxckout_set instead)

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>sync_clkout</i>	[IN] Recovered clock to be enabled or disabled.

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR on error.

8.30.5.16 vtss_phy_10g_xfp_clkout_get()

```
vtss_rc vtss_phy_10g_xfp_clkout_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    BOOL *const xfp_clkout )
```

Get the status of RXCLKOUT/TXCLKOUT from PHY. (recommended to use vtss_phy_10g_txckout_get instead)

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>xfp_clkout</i>	[IN] XFP clock configuration.

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR on error.

8.30.5.17 vtss_phy_10g_xfp_clkout_set()

```
vtss_rc vtss_phy_10g_xfp_clkout_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const BOOL xfp_clkout )
```

Enable or Disable the RXCLKOUT/TXCLKOUT from PHY. (recommended to use vtss_phy_10g_txckout_set instead)

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>xfp_clkout</i>	[IN] XFP clock to be enabled or disabled.

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR on error.

8.30.5.18 vtss_phy_10g_rxckout_get()

```
vtss_rc vtss_phy_10g_rxckout_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_10g_rxckout_conf_t *const rxckout )
```

Get the rx recovered clock output configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>rxckout</i>	[OUT] RXCKOUT clock configuration.

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR on error.

8.30.5.19 vtss_phy_10g_rxckout_set()

```
vtss_rc vtss_phy_10g_rxckout_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_10g_rxckout_conf_t *const rxckout )
```

Set the rx recovered clock output configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>rxckout</i>	[IN] RXCKOUT clock configuration.

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR on error.

8.30.5.20 vtss_phy_10g_txckout_get()

```
vtss_rc vtss_phy_10g_txckout_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_10g_txckout_conf_t *const txckout )
```

Get the status of tx recovered clock output configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>txckout</i>	[OUT] TXCKOUT clock configuration.

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR on error.

8.30.5.21 vtss_phy_10g_txckout_set()

```
vtss_rc vtss_phy_10g_txckout_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_10g_txckout_conf_t *const txckout )
```

Set the tx recovered clock output configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>txckout</i>	[IN] TXCKOUT clock configuration.

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR on error.

8.30.5.22 vtss_phy_10g_srefclk_conf_get()

```
vtss_rc vtss_phy_10g_srefclk_conf_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_10g_srefclk_mode_t *const srefclk )
```

Get the configuration of srefclk setting

Available for PHY family VENICE

This function should not be used any more, instead use the API function vtss_phy_10g_mode_get, see the parameter documentation for that function.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>srefclk</i>	[OUT] srefclk configuration.

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR on error.

8.30.5.23 vtss_phy_10g_srefclk_conf_set()

```
vtss_rc vtss_phy_10g_srefclk_conf_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_10g_srefclk_mode_t *const srefclk )
```

Set the configuration of srefclk setting. Available for PHY family VENICE

This function should not be used any more, instead use the API function vtss_phy_10g_mode_set, see the parameter documentation for that function.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>srefclk</i>	[IN] srefclk configuration.

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR on error.

8.30.5.24 vtss_phy_10g_sckout_conf_set()

```
vtss_rc vtss_phy_10g_sckout_conf_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_10g_sckout_conf_t *const sckout )
```

Set the configuration of sckout setting. Available for PHY family MALIBU

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>sckout</i>	[IN] sckout configuration.

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR on error.

8.30.5.25 vtss_phy_10g_ckout_conf_set()

```
vtss_rc vtss_phy_10g_ckout_conf_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_10g_ckout_conf_t *const ckout )
```

Set the configuration of ckout setting. Available for PHY family MALIBU

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>ckout</i>	[IN] ckout configuration.

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR on error.

8.30.5.26 vtss_phy_10g_line_clk_conf_set()

```
vtss_rc vtss_phy_10g_line_clk_conf_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_10g_line_clk_conf_t *const line_clk )
```

Set the configuration of sckout setting. Available for PHY family MALIBU

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>line_clk</i>	[IN] line_clk configuration.

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR on error.

8.30.5.27 vtss_phy_10g_host_clk_conf_set()

```
vtss_rc vtss_phy_10g_host_clk_conf_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_10g_host_clk_conf_t *const host_clk )
```

Set the configuration of sckout setting. Available for PHY family MALIBU

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>host_clk</i>	[IN] host_clk configuration.

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR on error.

8.30.5.28 vtss_phy_10g_line_recvrd_clk_conf_set()

```
vtss_rc vtss_phy_10g_line_recvrd_clk_conf_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_10g_line_clk_conf_t *const line_clk )
```

Set the configuration of line clk recovered setting. Available for PHY family MALIBU

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>line_clk</i>	[IN] line_recvrd_clk configuration.

Returns

VTSS_RC_OK on success.
VTSS_RC_ERROR on error.

8.30.5.29 vtss_phy_10g_host_recvrd_clk_conf_set()

```
vtss_rc vtss_phy_10g_host_recvrd_clk_conf_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_10g_host_clk_conf_t *const host_clk )
```

Set the configuration of host clk recovered setting. Available for PHY family MALIBU

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>host_clk</i>	[IN] host_clk configuration.

Returns

VTSS_RC_OK on success.
VTSS_RC_ERROR on error.

8.30.5.30 vtss_phy_10g_lane_sync_set()

```
vtss_rc vtss_phy_10g_lane_sync_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_10g_lane_sync_conf_t *const lane_sync )
```

Set the configuration of lane sync setting. Available for PHY family MALIBU

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>lane_sync</i>	[IN] ckout configuration.

Returns

VTSS_RC_OK on success.
VTSS_RC_ERROR on error.

8.30.5.31 vtss_phy_10g_debug_register_dump()

```
vtss_rc vtss_phy_10g_debug_register_dump (
    const vtss_inst_t inst,
    const vtss_debug_printf_t pr,
    BOOL clear,
    const vtss_port_no_t port_no )
```

Set the configuration of 10G PHY Dump setting. Available for PHY family Venice & Malibu

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>pr</i>	[IN] Print function.
<i>clear</i>	[IN] set for clearing the counters
<i>port_no</i>	[IN] Port number.

Returns

VTSS_RC_OK on success.
VTSS_RC_ERROR on error.

8.30.5.32 vtss_phy_10g_base_kr_train_aneg_get()

```
vtss_rc vtss_phy_10g_base_kr_train_aneg_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_10g_base_kr_train_aneg_t *const kr_tr_aneg )
```

Get the configuration of 10g_base_kr_tr_aneg setting. Available for PHY family Malibu,venice-c.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>kr_tr_aneg</i>	[OUT] 10g_base_kr_tr_aneg configuration.

Returns

VTSS_RC_OK on success.

VTSS_RC_ERROR on error.

8.30.5.33 vtss_phy_10g_base_kr_train_aneg_set()

```
vtss_rc vtss_phy_10g_base_kr_train_aneg_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_10g_base_kr_train_aneg_t *const kr_tr_aneg )
```

Set the configuration of 10g_base_kr_tr_aneg setting. Available for PHY family Malibu,venice-c.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>kr_tr_aneg</i>	[IN] 10g_base_kr_tr_aneg configuration.

Returns

VTSS_RC_OK on success.

VTSS_RC_ERROR on error.

8.30.5.34 vtss_phy_10g_base_host_kr_train_aneg_set()

```
vtss_rc vtss_phy_10g_base_host_kr_train_aneg_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_10g_base_kr_train_aneg_t *const kr_tr_aneg )
```

Set the configuration of Host 10g_base_kr_tr_aneg setting. Available for PHY family Malibu.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>kr_tr_aneg</i>	[IN] 10g_base_kr_tr_aneg configuration.

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR on error.

8.30.5.35 vtss_phy_10g_base_kr_conf_get()

```
vtss_rc vtss_phy_10g_base_kr_conf_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_10g_base_kr_conf_t *const kr_conf )
```

Get the configuration of Line 10G output buffer setting. Available for PHY family VENICE-c,Malibu.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>kr_conf</i>	[OUT] 10G output buffer configuration.

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR on error.

8.30.5.36 vtss_phy_10g_base_kr_conf_set()

```
vtss_rc vtss_phy_10g_base_kr_conf_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_10g_base_kr_conf_t *const kr_conf )
```

Set the configuration of Line 10G output buffer. Available for PHY family VENICE-c,MALIBU: Note: The parameters cm1,c0, c1 have a range of -32..31. The Output signal from the KR circuit is symmetric, i.e. the voltage output is configured value + 1/2lsb. (ex: -1 => -1/2lsb voltage level, 0 => +1/2lsb voltage level) The parameter ampl is set in steps of 25 mV, therefore the setting is rounded up to a multiplum og 25 mV, i.e. 1101..1125 => 1125 mVppd.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>kr_conf</i>	[IN] 10G output buffer configuration.

Generated by Doxygen

Returns

VTSS_RC_OK on success.

VTSS_RC_ERR_KR_CONF_NOT_SUPPORTED if the PHY on the port does not support 10GBASE_KR configuration
 VTSS_RC_ERR_KR_CONF_INVALID_PARAMETER if one of the parameter values are invalid
 or ($|cm1| + |c0| + |c1| > 31$)
 VTSS_RC_ERROR on other errors.

8.30.5.37 vtss_phy_10g_base_kr_host_conf_get()

```
vtss_rc vtss_phy_10g_base_kr_host_conf_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_10g_base_kr_conf_t *const kr_conf )
```

Get the configuration of Host 10G output buffer.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>kr_conf</i>	[OUT] host 10G output buffer configuration.

Returns

VTSS_RC_OK on success.

VTSS_RC_ERROR on error.

8.30.5.38 vtss_phy_10g_base_kr_host_conf_set()

```
vtss_rc vtss_phy_10g_base_kr_host_conf_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_10g_base_kr_conf_t *const kr_conf )
```

Set the configuration of host 10G output buffer. Note: The parameters cm1,c0, c1 have a range of -32..31. The Output signal from the KR circuit is symmetric, i.e. the voltage output is configured value + 1/2lsb. (ex: -1 => -1/2lsb voltage level, 0 => +1/2lsb voltage level) The parameter ampl is set in steps of 25 mV, therefore the setting is rounded up to a multiplum og 25 mV, i.e. 1101..1125 => 1125 mVppd.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>kr_conf</i>	[IN] host 10G output buffer configuration.

Returns

VTSS_RC_OK on success.

VTSS_RC_ERR_KR_CONF_NOT_SUPPORTED if the PHY on the port does not support 10GBASE_KR configuration
 VTSS_RC_ERR_KR_CONF_INVALID_PARAMETER if one of the parameter values are invalid
 or ($|cm1| + |c0| + |c1| > 31$)
 VTSS_RC_ERROR on other errors.

8.30.5.39 vtss_phy_10g_kr_status_get()

```
vtss_rc vtss_phy_10g_kr_status_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    BOOL direction,
    vtss_phy_10g_base_kr_status_t *const kr_status )
```

Get status of KR autonegotiation and training. Available for PHY family Malibu,Venice-c.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>direction</i>	[IN] Direction line or host.
<i>kr_status</i>	[OUT] 10g_base_kr status(aneg,trainging,fec).

Returns

VTSS_RC_OK on success.

VTSS_RC_ERROR on error.

8.30.5.40 vtss_phy_10g_ob_status_get()

```
vtss_rc vtss_phy_10g_ob_status_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_10g_ob_status_t *const ob_status )
```

Get the status of Output Buffer.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>ob_status</i>	[OUT] Status of output buffer

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR on error.

8.30.5.41 vtss_phy_10g_status_get()

```
vtss_rc vtss_phy_10g_status_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_10g_status_t *const status )
```

Get the link and fault status of the PHY sublayers.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>status</i>	[IN] Status of all sublayers

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR on error.

8.30.5.42 vtss_phy_10g_serdes_status_get()

```
vtss_rc vtss_phy_10g_serdes_status_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_10g_serdes_status_t *const status )
```

Get the status of PHY including sub layers.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>status</i>	[IN] Status of PLL,SUB layers

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR on error.

8.30.5.43 vtss_phy_10g_reset()

```
vtss_rc vtss_phy_10g_reset (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no )
```

Reset the phy. Phy is reset to default values.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.

Returns

VTSS_RC_OK on success.

VTSS_RC_ERROR on error.

8.30.5.44 vtss_phy_10g_clause_37_status_get()

```
vtss_rc vtss_phy_10g_clause_37_status_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_10g_clause_37_cmn_status_t *const status )
```

Get clause 37 status.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>status</i>	[OUT] Clause 37 status of the line and host link.

Returns

VTSS_RC_OK on success.

VTSS_RC_ERROR on error.

8.30.5.45 vtss_phy_10g_clause_37_control_get()

```
vtss_rc vtss_phy_10g_clause_37_control_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_10g_clause_37_control_t *const control )
```

Get clause 37 control configuration from software.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>control</i>	[OUT] Clause 37 configuration,control.line,control.host are 'in' parameters.

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR on error.

8.30.5.46 vtss_phy_10g_clause_37_control_set()

```
vtss_rc vtss_phy_10g_clause_37_control_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_10g_clause_37_control_t *const control )
```

Set clause 37 control configuration.

Clause 37 can be configured independently on HOST,LINE 1G PCSs 1G speed is only supported in 1000-X aneg

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>control</i>	[OUT] Clause 37 configuration. Same configuration is applied to Host and Line interface.

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR on error.

8.30.5.47 vtss_phy_10g_loopback_set()

```
vtss_rc vtss_phy_10g_loopback_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_10g_loopback_t *const loopback )
```

Enable/Disable a phy network or system loopback.

Only one loopback mode can be active at the same time.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>loopback</i>	[IN] Loopback settings. When disabling a loopback, the lb_type is ignored, i.e. the active loopback is disabled.

Returns

VTSS_RC_OK on success.

VTSS_RC_ERROR on error.

Error conditions: Loopback not supported for the PHY Attempt to enable loopback while loopback is already active Attempt to disable loopback while no loopback is active

8.30.5.48 vtss_phy_10g_loopback_get()

```
vtss_rc vtss_phy_10g_loopback_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_10g_loopback_t *const loopback )
```

Get loopback settings.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>loopback</i>	[OUT] Current loopback settings.

Returns

VTSS_RC_OK on success.

VTSS_RC_ERROR on error.

8.30.5.49 vtss_phy_10g_cnt_get()

```
vtss_rc vtss_phy_10g_cnt_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_10g_cnt_t *const cnt )
```

Get counters.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>cnt</i>	[OUT] Phy counters

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR on error.

8.30.5.50 vtss_phy_10g_power_get()

```
vtss_rc vtss_phy_10g_power_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_10g_power_t *const power )
```

Get the power settings.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>power</i>	[OUT] power settings

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR on error.

8.30.5.51 vtss_phy_10g_power_set()

```
vtss_rc vtss_phy_10g_power_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_10g_power_t *const power )
```

Set the power settings.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>power</i>	[IN] power settings

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR on error.

8.30.5.52 vtss_phy_10G_is_valid()

```
BOOL vtss_phy_10G_is_valid (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no )
```

Gives a True/False value if the Phy is supported by the API
 Only Vitesse phys are supported. [vtss_phy_10g_mode_set\(\)](#) must be applied.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.

Returns

TRUE : Phy is supported.
 FALSE : Phy is not supported.

8.30.5.53 vtss_phy_10g_failover_set()

```
vtss_rc vtss_phy_10g_failover_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_10g_failover_mode_t *const mode )
```

Set the failover mode.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number. (Use any port within the phy).
<i>mode</i>	[IN] Failover mode

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR on error.

8.30.5.54 vtss_phy_10g_failover_get()

```
vtss_rc vtss_phy_10g_failover_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_10g_failover_mode_t *const mode )
```

Get the failover mode.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>mode</i>	[OUT] failover mode

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR on error.

8.30.5.55 vtss_phy_10g_auto_failover_set()

```
vtss_rc vtss_phy_10g_auto_failover_set (
    const vtss_inst_t inst,
    vtss_phy_10g_auto_failover_conf_t *const mode )
```

Set the automatic failover mode.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>mode</i>	[IN] Automatic Failover mode

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR on error.

8.30.5.56 vtss_phy_10g_auto_failover_get()

```
vtss_rc vtss_phy_10g_auto_failover_get (
    const vtss_inst_t inst,
    vtss_phy_10g_auto_failover_conf_t *const mode )
```

Get the Automatic failover mode Configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>mode</i>	[OUT] failover mode

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR on error.

8.30.5.57 vtss_phy_10g_vscope_conf_set()

```
vtss_rc vtss_phy_10g_vscope_conf_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_10g_vscope_conf_t *const conf )
```

set VSCOPE fast scan configuration

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>conf</i>	[IN] VSCOPE fast scan configuration

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR on error.

8.30.5.58 vtss_phy_10g_vscope_conf_get()

```
vtss_rc vtss_phy_10g_vscope_conf_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_10g_vscope_conf_t *const conf )
```

get VSCOPE fast scan configuration

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>conf</i>	[OUT] VSCOPE fast scan configuration

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR on error.

8.30.5.59 vtss_phy_10g_vscope_scan_status_get()

```
vtss_rc vtss_phy_10g_vscope_scan_status_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_10g_vscope_scan_status_t *const conf )
```

set VSCOPE fast scan configuration

\ brief VSCOPE fast scan status

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>conf</i>	[IN] VSCOPE fast scan configuration

Returns

VTSS_RC_OK on success.

VTSS_RC_ERROR on error.

8.30.5.60 vtss_phy_10g_pcs_prbs_gen_conf_set()

```
vtss_rc vtss_phy_10g_pcs_prbs_gen_conf_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_10g_pcs_prbs_gen_conf_t *const conf,
    const BOOL line )
```

Set PCS-prbs generator Configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>conf</i>	[IN] Pcs-Prbs generator configuration
<i>line</i>	[IN] Direction

Returns

VTSS_RC_OK on success.

VTSS_RC_ERROR on error.

8.30.5.61 vtss_phy_10g_pcs_prbs_gen_conf_get()

```
vtss_rc vtss_phy_10g_pcs_prbs_gen_conf_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_10g_pcs_prbs_gen_conf_t *const conf,
    const BOOL line )
```

Get PCS-prbs generator Configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>conf</i>	[OUT] Pcs-Prbs generator configuration
<i>line</i>	[IN] Direction

Returns

VTSS_RC_OK on success.

VTSS_RC_ERROR on error.

8.30.5.62 vtss_phy_10g_pcs_prbs_mon_conf_set()

```
vtss_rc vtss_phy_10g_pcs_prbs_mon_conf_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_10g_pcs_prbs_mon_conf_t *const conf,
    const BOOL line )
```

Set PCS-prbs monitor Configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>conf</i>	[IN] Pcs-Prbs monitor configuration
<i>line</i>	[IN] Direction

Returns

VTSS_RC_OK on success.

VTSS_RC_ERROR on error.

8.30.5.63 vtss_phy_10g_pcs_prbs_mon_conf_get()

```
vtss_rc vtss_phy_10g_pcs_prbs_mon_conf_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_10g_pcs_prbs_mon_conf_t *const conf,
    const BOOL line )
```

Get PCS-prbs monitor Configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>conf</i>	[OUT] Pcs-Prbs monitor configuration
<i>line</i>	[IN] Direction

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR on error.

8.30.5.64 vtss_phy_10g_pcs_prbs_mon_status_get()

```
vtss_rc vtss_phy_10g_pcs_prbs_mon_status_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_10g_pcs_prbs_mon_conf_t *const conf,
    const BOOL line )
```

Get PCS-prbs monitor status.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>conf</i>	[OUT] Pcs-Prbs monitor status
<i>line</i>	[IN] Direction

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR on error.

8.30.5.65 vtss_phy_10g_prbs_gen_conf()

```
vtss_rc vtss_phy_10g_prbs_gen_conf (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_10g_prbs_gen_conf_t *const conf )
```

set prbs generator Configuration

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>conf</i>	[IN] Prbs configuration

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR on error.

8.30.5.66 vtss_phy_10g_prbs_gen_conf_get()

```
vtss_rc vtss_phy_10g_prbs_gen_conf_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_10g_prbs_gen_conf_t *const conf,
    BOOL line )
```

get prbs generator Configuration

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>conf</i>	[OUT] Prbs configuration
<i>line</i>	[IN] Direction in which Prbs generator configuration is requested

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR on error.

8.30.5.67 vtss_phy_10g_prbs_mon_conf()

```
vtss_rc vtss_phy_10g_prbs_mon_conf (
    const vtss_inst_t inst,
```

```
const vtss_port_no_t port_no,
vtss_phy_10g_prbs_mon_conf_t *const conf )
```

set prbs generator Configuration

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>conf</i>	[IN] Prbs Monitor configuration

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR on error.

8.30.5.68 vtss_phy_10g_prbs_mon_conf_get()

```
vtss_rc vtss_phy_10g_prbs_mon_conf_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_10g_prbs_mon_conf_t *const conf,
    BOOL line )
```

prbs generator Configuration get

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>conf</i>	[IN] Prbs Monitor configuration
<i>line</i>	[IN] Direction in which Prbs Monitor configuration is requested

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR on error.

8.30.5.69 vtss_phy_10g_prbs_mon_status_get()

```
vtss_rc vtss_phy_10g_prbs_mon_status_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_10g_prbs_mon_conf_t *const mon_status,
    BOOL line,
    BOOL reset )
```

prbs Checker Status get

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>mon_status</i>	[OUT] Prbs Monitor status
<i>line</i>	[IN] Direction in which Prbs Monitor status is requested
<i>reset</i>	[IN] Resets prbs counters before retrieving the status

Returns

VTSS_RC_OK on success.

VTSS_RC_ERROR on error.

8.30.5.70 vtss_phy_10g_pkt_gen_conf()

```
vtss_rc vtss_phy_10g_pkt_gen_conf (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_10g_pkt_gen_conf_t *const conf )
```

Set Packet generation Configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>conf</i>	[IN] Packet generator configuration

Returns

VTSS_RC_OK on success.

VTSS_RC_ERROR on error.

8.30.5.71 vtss_phy_10g_pkt_mon_conf()

```
vtss_rc vtss_phy_10g_pkt_mon_conf (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    BOOL ts_rd,
    vtss_phy_10g_pkt_mon_conf_t *const conf,
    vtss_phy_10g_timestamp_val_t *const conf_ts )
```

Set Packet Monitor Configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>ts_rd</i>	[IN] Flag to indicate that timestamp fifo is also to be read.
<i>conf</i>	Packet monitor configuration
<i>conf</i> ← <i>ts</i>	[OUT] Timestamp value array.

Returns

VTSS_RC_OK on success.

VTSS_RC_ERROR on error.

8.30.5.72 vtss_phy_10g_pkt_mon_counters_get()

```
vtss_rc vtss_phy_10g_pkt_mon_counters_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_10g_pkt_mon_conf_t *const conf )
```

Set/Get Packet mon Counters.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>conf</i>	Packet monitor configuration

Returns

VTSS_RC_OK on success.

VTSS_RC_ERROR on error.

8.30.5.73 vtss_phy_10g_id_get()

```
vtss_rc vtss_phy_10g_id_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_10g_id_t *const phy_id )
```

Read the Phy Id.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>phy_id</i>	[OUT] The part number and revision.

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR on error.

8.30.5.74 vtss_phy_10g_gpio_mode_set()

```
vtss_rc vtss_phy_10g_gpio_mode_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_gpio_10g_no_t gpio_no,
    const vtss_gpio_10g_gpio_mode_t *const mode )
```

Set GPIO mode. There is only one set of GPIO per PHY chip - not per port.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number that identify the PHY chip.
<i>gpio_no</i>	[IN] GPIO pin number < VTSS_10G_PHY_GPIO_MAX.
<i>mode</i>	[IN] GPIO mode.

Returns

Return code.

8.30.5.75 vtss_phy_10g_gpio_mode_get()

```
vtss_rc vtss_phy_10g_gpio_mode_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_gpio_10g_no_t gpio_no,
    vtss_gpio_10g_gpio_mode_t *const mode )
```

Get GPIO mode.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number that identify the PHY chip.
<i>gpio_no</i>	[IN] GPIO pin number.
<i>mode</i>	[OUT] GPIO mode.

Generated by Doxygen

Returns

Return code.

8.30.5.76 vtss_phy_10g_gpio_read()

```
vtss_rc vtss_phy_10g_gpio_read (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_gpio_10g_no_t gpio_no,
    BOOL *const value )
```

Read from GPIO input pin.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>gpio_no</i>	[IN] GPIO pin number.
<i>value</i>	[OUT] TRUE if pin is high, FALSE if it is low.

Returns

Return code.

8.30.5.77 vtss_phy_10g_gpio_write()

```
vtss_rc vtss_phy_10g_gpio_write (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_gpio_10g_no_t gpio_no,
    const BOOL value )
```

Write to GPIO output pin.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>gpio_no</i>	[IN] GPIO pin number.
<i>value</i>	[IN] TRUE to set pin high, FALSE to set pin low.

Returns

Return code.

8.30.5.78 vtss_phy_10g_event_enable_set()

```
vtss_rc vtss_phy_10g_event_enable_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_10g_event_t ev_mask,
    const BOOL enable )
```

Enabling / Disabling of events.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>ev_mask</i>	[IN] Mask containing events that are enabled/disabled
<i>enable</i>	[IN] Enable/disable of event

Returns

Return code.

8.30.5.79 vtss_phy_10g_event_enable_get()

```
vtss_rc vtss_phy_10g_event_enable_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_10g_event_t *const ev_mask )
```

Get Enabling of events.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>ev_mask</i>	[OUT] Mask containing events that are enabled

Returns

Return code.

8.30.5.80 vtss_phy_10g_extended_event_enable_get()

```
vtss_rc vtss_phy_10g_extended_event_enable_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_10g_extnd_event_t *const ex_ev_mask )
```

Get Enabling of events.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>ex_ev_mask</i>	[OUT] Mask containing extended events that are enabled

Returns

Return code.

8.30.5.81 vtss_phy_10g_event_poll()

```
vtss_rc vtss_phy_10g_event_poll (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_10g_event_t *const ev_mask )
```

Polling for active events.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>ev_mask</i>	[OUT] Mask containing events that are active

Returns

Return code.

8.30.5.82 vtss_phy_10g_pcs_status_get()

```
vtss_rc vtss_phy_10g_pcs_status_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_10g_extnd_event_t *const ex_events )
```

poll and clear PCS STICKY Register

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>ex_events</i>	[OUT] Event mask containing events that are active

Returns

Return code.

8.30.5.83 vtss_phy_10g_extended_event_poll()

```
vtss_rc vtss_phy_10g_extended_event_poll (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_10g_extnd_event_t *const ex_events )
```

Polling for active events.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>ex_events</i>	[OUT] Event mask containing events that are active

Returns

Return code.

8.30.5.84 vtss_phy_10g_extended_event_enable_set()

```
vtss_rc vtss_phy_10g_extended_event_enable_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_10g_extnd_event_t ex_ev_mask,
    const BOOL extnd_enable )
```

Enabling / Disabling of events.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>ex_ev_mask</i>	[IN] Mask containing exetnded events that are enabled/disabled
<i>extnd_enable</i>	[IN] Enable/disable of event

Returns

Return code.

8.30.5.85 vtss_phy_10g_poll_1sec()

```
vtss_rc vtss_phy_10g_poll_1sec (
    const vtss_inst_t inst )
```

Function is called once a second.

Parameters

<i>inst</i>	[IN] Target instance reference.
-------------	---------------------------------

Returns

Return code.

8.30.5.86 vtss_phy_10g_edc_fw_status_get()

```
vtss_rc vtss_phy_10g_edc_fw_status_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_10g_fw_status_t *const status )
```

Internal microprocessor status.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>status</i>	[OUT] Status of the EDC FW running on the internal CPU

Returns

Return code.

8.30.5.87 vtss_phy_10g_fc_buffer_reset()

```
vtss_rc vtss_phy_10g_fc_buffer_reset (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no )
```

debug function for PHY 10G FC buffer reset

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Any phy port with the chip

Returns

VTSS_RC_OK - success of fc buffer reset

8.30.5.88 vtss_phy_10g_csr_read()

```
vtss_rc vtss_phy_10g_csr_read (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const u32 dev,
    const u32 addr,
    u32 *const value )
```

CSR register read.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>dev</i>	[IN] Device id (or MMD)
<i>addr</i>	[IN] Addr of the register, 16 or 32 bit
<i>value</i>	[OUT] Return value of the register

Returns

Return code.

8.30.5.89 vtss_phy_10g_csr_write()

```
vtss_rc vtss_phy_10g_csr_write (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const u32 dev,
    const u32 addr,
    const u32 value )
```

CSR register write.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>dev</i>	[IN] Device id (or MMD)
<i>addr</i>	[IN] Addr of the register, 16 or 32 bit
<i>value</i>	[IN] Value to be written

Returns

Return code.

8.30.5.90 vtss_phy_warm_start_10g_failed_get()

```
vtss_rc vtss_phy_warm_start_10g_failed_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no )
```

Function for checking if any issue were seen during warm-start.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] The port in question.

Returns

Return code. VTSS_RC_OK if no errors were seen during warm-start else VTSS_RC_ERROR.

8.30.5.91 vtss_phy_10g_sgmii_mode_set()

```
vtss_rc vtss_phy_10g_sgmii_mode_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    BOOL enable )
```

Enables Pass through mode in 10G PHY.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>enable</i>	[IN] Enables SGMII mode.

Returns

VTSS_RC_OK on success.
VTSS_RC_ERROR on error.

8.30.5.92 vtss_phy_10g_i2c_read()

```
vtss_rc vtss_phy_10g_i2c_read (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const u16 addr,
    u16 * value )
```

read from i2c device

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>addr</i>	[IN] Addr of the SFP ROM register
<i>value</i>	[OUT] Return Value of the register

Returns

Return code.

8.30.5.93 vtss_phy_10g_i2c_write()

```
vtss_rc vtss_phy_10g_i2c_write (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const u16 addr,
    const u16 * value )
```

Write to i2c device.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>addr</i>	[IN] Addr of the SFP ROM register
<i>value</i>	[IN] value to be written to register

Returns

Return code.

8.30.5.94 vtss_phy_10g_get_user_data()

```
vtss_rc vtss_phy_10g_get_user_data (
    const vtss_inst_t inst,
```

```
const vtss_port_no_t port_no,
void ** user_data )
```

Gets generic pointer in vtss_state structure.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>user_data</i>	[OUT] Gets value in generic pointer

Returns

Return code.

8.31 vtss_api/include/vtss_phy_api.h File Reference

PHY API.

```
#include <vtss/api/types.h>
#include <vtss/api/port.h>
#include <vtss/api/phy.h>
#include <vtss_misc_api.h>
```

Data Structures

- struct [vtss_phy_led_mode_select_t](#)
LED model selection.
- struct [vtss_phy_type_t](#)
Phy type information.
- struct [vtss_phy_rgmii_conf_t](#)
PHY RGMII configuration.
- struct [vtss_phy_tbi_conf_t](#)
PHY TBI configuration.
- struct [vtss_phy_reset_conf_t](#)
PHY reset structure.
- struct [vtss_phy_forced_t](#)
PHY forced mode configuration.
- struct [vtss_phy_aneg_t](#)
PHY auto negotiation advertisement.
- struct [vtss_phy_mac_serdes_pcs_ctrl_t](#)
PHY MAC SerDes PCS Control, Reg16E3.
- struct [vtss_phy_media_serdes_pcs_ctrl_t](#)
PHY MEDIA SerDes PCS Control, Reg23E3.
- struct [vtss_phy_conf_t](#)
PHY configuration.
- struct [vtss_phy_conf_1g_t](#)
PHY 1G configuration.

- struct `vtss_phy_status_1g_t`
PHY 1G status.
- struct `vtss_phy_power_conf_t`
PHY power configuration.
- struct `vtss_phy_power_status_t`
PHY power status.
- struct `vtss_phy_clock_conf_t`
PHY clock configuration.
- struct `vtss_phy_veriphy_result_t`
VeriPHY result.
- struct `vtss_phy_eee_conf_t`
EEE configuration.
- struct `vtss_phy_enhanced_led_control_t`
enhanced LED control
- struct `vtss_phy_statistic_t`
Phy statistic information.
- struct `vtss_phy_loopback_t`
1G Phy loopbacks
- struct `vtss_wol_mac_addr_t`
Structure for Wake-On-LAN MAC Address.
- struct `vtss_secure_on_passwd_t`
Structure for Wake-On-LAN Secure-On Password.
- struct `vtss_phy_wol_conf_t`
Structure for Get/Set Wake-On-LAN configuration.
- struct `vtss_rcpll_status_t`
Structure for Get PHY RC-PLL status.
- struct `vtss_lcpll_status_t`
Structure for Get PHY LC-PLL status.

Macros

- #define MAX_CFG_BUF_SIZE 38
- #define MAX_STAT_BUF_SIZE 8
- #define VTSS_PHY_POWER_ACTIPHYS_BIT 0
- #define VTSS_PHY_POWER_DYNAMIC_BIT 1
- #define VTSS_PHY_ACTIPHYS_PWR 100
- #define VTSS_PHY_LINK_DOWN_PWR 200
- #define VTSS_PHY_LINK_UP_FULL_PWR 400
- #define VTSS_PHY_RECov_CLK1 0
PHY active clock out.
- #define VTSS_PHY_RECov_CLK2 1
- #define VTSS_PHY_RECov_CLK_NUM 2
- #define VTSS_PHY_PAGE_STANDARD 0x0000
- #define VTSS_PHY_PAGE_EXTENDED 0x0001
- #define VTSS_PHY_PAGE_EXTENDED_2 0x0002
- #define VTSS_PHY_PAGE_EXTENDED_3 0x0003
- #define VTSS_PHY_PAGE_EXTENDED_4 0x0004
- #define VTSS_PHY_PAGE_GPIO 0x0010
- #define VTSS_PHY_PAGE_1588 0x1588
- #define VTSS_PHY_PAGE_MACSEC 0x0004
- #define VTSS_PHY_PAGE_TEST 0x2A30

- #define VTSS_PHY_PAGE_TR 0x52B5
- #define VTSS_PHY_PAGE_0x2DAF 0x2DAF
- #define VTSS_PHY_REG_STANDARD (VTSS_PHY_PAGE_STANDARD<<5)
- #define VTSS_PHY_REG_EXTENDED (VTSS_PHY_PAGE_EXTENDED<<5)
- #define VTSS_PHY_REG_GPIO (VTSS_PHY_PAGE_GPIO<<5)
- #define VTSS_PHY_REG_TEST (VTSS_PHY_PAGE_TEST<<5)
- #define VTSS_PHY_REG_TR (VTSS_PHY_PAGE_TR<<5)
- #define VTSS_PHY_LINK_LOS_EV (1 << 0)
- #define VTSS_PHY_LINK_FFAIL_EV (1 << 1)
- #define VTSS_PHY_LINK_AMS_EV (1 << 2)
- #define VTSS_PHY_LINK_SPEED_STATE_CHANGE_EV (1 << 3)
- #define VTSS_PHY_LINK_FDX_STATE_CHANGE_EV (1 << 4)
- #define VTSS_PHY_LINK_AUTO_NEG_ERROR_EV (1 << 5)
- #define VTSS_PHY_LINK_AUTO_NEG_COMPLETE_EV (1 << 6)
- #define VTSS_PHY_LINK_INLINE_POW_DEV_DETECT_EV (1 << 7)
- #define VTSS_PHY_LINK_SYMBOL_ERR_INT_EV (1 << 8)
- #define VTSS_PHY_LINK_TX_FIFO_OVERFLOW_INT_EV (1 << 9)
- #define VTSS_PHY_LINK_RX_FIFO_OVERFLOW_INT_EV (1 << 10)
- #define VTSS_PHY_LINK_FALSE_CARRIER_INT_EV (1 << 11)
- #define VTSS_PHY_LINK_LINK_SPEED_DS_DETECT_EV (1 << 12)
- #define VTSS_PHY_LINK_MASTER_SLAVE_RES_ERR_EV (1 << 13)
- #define VTSS_PHY_LINK_RX_ER_INT_EV (1 << 14)
- #define VTSS_PHY_LINK_EXTENDED_REG_INT_EV (1 << 15)
- #define VTSS_PHY_LINK_WAKE_ON_LAN_INT_EV (1 << 16)
- #define VTSS_PHY_LINK_EXT_EEE_WAKE_ERR_EV (1 << 17)
- #define VTSS_PHY_LINK_EXT_EEE_WAIT_TS_EV (1 << 18)
- #define VTSS_PHY_LINK_EXT_EEE_WAIT_RX_TQ_EV (1 << 19)
- #define VTSS_PHY_LINK_EXT_EEE_LINKFAIL_EV (1 << 20)
- #define VTSS_PHY_LINK_EXT_RR_SW_COMPL_EV (1 << 21)
- #define VTSS_PHY_LINK_EXT_MACSEC_HOST_MAC_EV (1 << 22)
- #define VTSS_PHY_LINK_EXT_MACSEC_LINE_MAC_EV (1 << 23)
- #define VTSS_PHY_LINK_EXT_MACSEC_FC_BUFF_EV (1 << 24)
- #define VTSS_PHY_LINK_EXT_MACSEC_INGRESS_EV (1 << 25)
- #define VTSS_PHY_LINK_EXT_MACSEC_EGRESS_EV (1 << 26)
- #define VTSS_PHY_LINK_EXT_MEM_INT_RING_EV (1 << 27)
- #define MAX_WOL_MAC_ADDR_SIZE 6
- #define MAX_WOL_PASSWD_SIZE 6
- #define MIN_WOL_PASSWD_SIZE 4

Typedefs

- typedef u16 vtss_phy_recov_clk_t
- typedef u8 vtss_phy_led_intensity

PHY led intensity.
- typedef u32 vtss_phy_event_t

PHY interrupt event type.

Enumerations

- enum `vtss_phy_part_number_t` {

`VTSS_PHY_TYPE_NONE` = 0, `VTSS_PHY_TYPE_8201` = 8201, `VTSS_PHY_TYPE_8204` = 8204, `VTSS_PHY_TYPE_8211` = 8211,
 `VTSS_PHY_TYPE_8221` = 8221, `VTSS_PHY_TYPE_8224` = 8224, `VTSS_PHY_TYPE_8234` = 8234, `VTSS_PHY_TYPE_8244` = 8244,
 `VTSS_PHY_TYPE_8538` = 8538, `VTSS_PHY_TYPE_8558` = 8558, `VTSS_PHY_TYPE_8574` = 8574, `VTSS_PHY_TYPE_8504` = 8504,
 `VTSS_PHY_TYPE_8572` = 8572, `VTSS_PHY_TYPE_8552` = 8552, `VTSS_PHY_TYPE_8501` = 8501, `VTSS_PHY_TYPE_8502` = 8502,
 `VTSS_PHY_TYPE_7435` = 7435, `VTSS_PHY_TYPE_8658` = 8658, `VTSS_PHY_TYPE_8601` = 8601, `VTSS_PHY_TYPE_8641` = 8641,
 `VTSS_PHY_TYPE_7385` = 7385, `VTSS_PHY_TYPE_7388` = 7388, `VTSS_PHY_TYPE_7389` = 7389, `VTSS_PHY_TYPE_7390` = 7390,
 `VTSS_PHY_TYPE_7395` = 7395, `VTSS_PHY_TYPE_7398` = 7398, `VTSS_PHY_TYPE_7500` = 7500, `VTSS_PHY_TYPE_7501` = 7501,
 `VTSS_PHY_TYPE_7502` = 7502, `VTSS_PHY_TYPE_7503` = 7503, `VTSS_PHY_TYPE_7504` = 7504, `VTSS_PHY_TYPE_7505` = 7505,
 `VTSS_PHY_TYPE_7506` = 7506, `VTSS_PHY_TYPE_7507` = 7507, `VTSS_PHY_TYPE_8634` = 8634, `VTSS_PHY_TYPE_8664` = 8664,
 `VTSS_PHY_TYPE_8512` = 8512, `VTSS_PHY_TYPE_8522` = 8522, `VTSS_PHY_TYPE_7420` = 7420, `VTSS_PHY_TYPE_8582` = 8582,
 `VTSS_PHY_TYPE_8584` = 8584, `VTSS_PHY_TYPE_8575` = 8575, `VTSS_PHY_TYPE_8564` = 8564, `VTSS_PHY_TYPE_8562` = 8562,
 `VTSS_PHY_TYPE_8586` = 8586, `VTSS_PHY_TYPE_8514` = 8514 }

PHY part ids supported.
- enum `vtss_phy_led_mode_t` {

`LINK_ACTIVITY`, `LINK100_ACTIVITY`, `LINK100_ACTIVITY`, `LINK10_ACTIVITY`,
 `LINK100_1000_ACTIVITY`, `LINK10_1000_ACTIVITY`, `LINK10_100_ACTIVITY`, `LINK100BASE_FX_1000BASE_X_ACTIVITY`,
 `DUPLEX_COLLISION`, `COLLISION_ACTIVITY`, `BASE100_FX_1000BASE_X_FIBER_ACTIVITY`,
 `AUTONEGOTIATION_FAULT`, `LINK100BASE_X_ACTIVITY`, `LINK100BASE_FX_ACTIVITY`, `BASE100_ACTIVITY`,
 `BASE100_FX_ACTIVITY`, `FORCE_LED_OFF`, `FORCE_LED_ON`, `FAST_LINK_FAIL` }

PHY LED modes.
- enum `vtss_phy_led_number_t` { `LED0`, `LED1`, `LED2`, `LED3` }

List of LED pins per port.
- enum `vtss_phy_media_interface_t` {

`VTSS_PHY_MEDIA_IF_CU`, `VTSS_PHY_MEDIA_IF_SFP_PASSTHRU`, `VTSS_PHY_MEDIA_IF_FL_1000BX`,
 `VTSS_PHY_MEDIA_IF_FL_100FX`,
 `VTSS_PHY_MEDIA_IF_AMS CU_PASSTHRU`, `VTSS_PHY_MEDIA_IF_AMS FI_PASSTHRU`, `VTSS_PHY_MEDIA_IF_AMS`
`VTSS_PHY_MEDIA_IF_AMS FI_1000BX`,
 `VTSS_PHY_MEDIA_IF_AMS CU_100FX`, `VTSS_PHY_MEDIA_IF_AMS FI_100FX` }

PHY media interface type.
- enum `vtss_phy_mdi_t` { `VTSS_PHY_MDIX_AUTO`, `VTSS_PHY_MDI`, `VTSS_PHY_MDIX` }

PHY media interface type.
- enum `rgmii_skew_delay_psec_t` {

`rgmii_skew_delay_200_psec` = 200, `rgmii_skew_delay_800_psec` = 800, `rgmii_skew_delay_1100_psec` =
 1100, `rgmii_skew_delay_1700_psec` = 1700,
 `rgmii_skew_delay_2000_psec` = 2000, `rgmii_skew_delay_2300_psec` = 2300, `rgmii_skew_delay_2600_psec` =
 2600, `rgmii_skew_delay_3400_psec` = 3400 }

RGMII skew values.
- enum `vtss_phy_forced_reset_t` { `VTSS_PHY_FORCE_RESET` = 0, `VTSS_PHY_NOFORCE_RESET` = 1 }

PHY forced reset interface type.
- enum `vtss_phy_pkt_mode_t` { `VTSS_PHY_PKT_MODE_IEEE_1_5_KB`, `VTSS_PHY_PKT_MODE_JUMBO_9_KB`,
 `VTSS_PHY_PKT_MODE_JUMBO_12_KB` }

PHY packet mode configuration.

- enum `vtss_phy_mode_t` { `VTSS_PHY_MODE_ANEG`, `VTSS_PHY_MODE_FORCED`, `VTSS_PHY_MODE_POWER_DOWN` }

PHY mode.

- enum `vtss_phy_fast_link_fail_t` { `VTSS_PHY_FAST_LINK_FAIL_ENABLE` = 1, `VTSS_PHY_FAST_LINK_FAIL_DISABLE` }

PHY fast link failure pin enable/disable.

- enum `vtss_phy_sigdet_polarity_t` { `VTSS_PHY_SIGDET_POLARITY_ACT_LOW` = 1, `VTSS_PHY_SIGDET_POLARITY_ACT_HIGH` }

PHY Sigdet pin polarity configuration.

- enum `vtss_phy_unidirectional_t` { `VTSS_PHY_UNIDIRECTIONAL_DISABLE` = 0, `VTSS_PHY_UNIDIRECTIONAL_ENABLE` }

PHY Unidirectional enable/disable.

- enum `vtss_phy_mac_serdes_pcs_sgmii_pre` { `VTSS_PHY_MAC_SERD_PCS_SGMII_IN_PRE_NONE` = 0, `VTSS_PHY_MAC_SERD_PCS_SGMII_IN_PRE_ONE` = 1, `VTSS_PHY_MAC_SERD_PCS_SGMII_IN_PRE_TWO` = 2, `VTSS_PHY_MAC_SERD_PCS_SGMII_IN_PRE_RSVD` = 3 }

PHY MEDIA SerDes PCS Remote Fault Indication, See Clause 37, Table 37-3.

- enum `vtss_phy_media_rem_fault_t` { `VTSS_PHY_MEDIA_SERD_PCS_Rem_Fault_No_Error` = 0, `VTSS_PHY_MEDIA_SERD_PCS_Rem_Fault_Offline` = 1, `VTSS_PHY_MEDIA_SERD_PCS_Rem_Fault_Link_Fail` = 2, `VTSS_PHY_MEDIA_SERD_PCS_Rem_Fault_ANEG_Error` = 3 }

PHY MEDIA SerDes PCS Remote Fault Indication, See Clause 37, Table 37-3.

- enum `vtss_phy_media_force_ams_sel_t` { `VTSS_PHY_MEDIA_FORCE_AMS_SELECTION_NORMAL` = 0, `VTSS_PHY_MEDIA_FORCE_AMS_SELECTION_SERDES` = 1, `VTSS_PHY_MEDIA_FORCE_AMS_SELECTION_COPPER` = 2 }

PHY AMS Force configuration.

- enum `vtss_phy_clk_source_t` { `VTSS_PHY_CLK_DISABLED`, `VTSS_PHY_SERDES_MEDIA`, `VTSS_PHY_COPPER_MEDIA`, `VTSS_PHY_TCLK_OUT`, `VTSS_PHY_LOCAL_XTAL` }

PHY clock sources.

- enum `vtss_phy_freq_t` { `VTSS_PHY_FREQ_25M`, `VTSS_PHY_FREQ_125M`, `VTSS_PHY_FREQ_3125M` }

PHY clock frequencies.

- enum `vtss_phy_clk_squelch` { `VTSS_PHY_CLK_SQUELCH_MAX` = 0, `VTSS_PHY_CLK_SQUELCH_MED` = 1, `VTSS_PHY_CLK_SQUELCH_MIN` = 2, `VTSS_PHY_CLK_SQUELCH_NONE` = 3 }

PHY clock squelch levels.

- enum `vtss_phy_gpio_mode_t` { `VTSS_PHY_GPIO_ALT_0` = 0, `VTSS_PHY_GPIO_ALT_1` = 1, `VTSS_PHY_GPIO_ALT_2` = 2, `VTSS_PHY_GPIO_OUT` = 3, `VTSS_PHY_GPIO_IN` = 4 }

GPIO pin operating mode.

- enum `vtss_eee_mode_t` { `EEE_DISABLE`, `EEE_ENABLE`, `EEE_REG_UPDATE` }

EEE mode.

- enum `lb_type` { `VTSS_LB_1G_NONE`, `VTSS_LB_FAR_END`, `VTSS_LB_NEAR_END` }

Internal loop-back type.

- enum `vtss_wol_passwd_len_type_t` { `VTSS_WOL_PASSWD_LEN_4` = 4, `VTSS_WOL_PASSWD_LEN_6` = 6 }

Structure for Wake-On-LAN Password Length configuration.

- enum `vtss_fefi_mode_t` { `VTSS_100FX_FEFI_NORMAL` = 0, `VTSS_100FX_FEFI_FORCE_SUPPRESS` = 2, `VTSS_100FX_FEFI_FORCE_ENABLE` = 3 }

Far-End Failure Indication modes These settings map to PHY Reg23E3, bits 0 and 1.

Functions

- `vtss_rc vtss_phy_pre_reset (const vtss_inst_t inst, const vtss_port_no_t port_no)`
Must be call previous to port PHY Reset (vtss_phy_reset).
- `vtss_rc vtss_phy_post_reset (const vtss_inst_t inst, const vtss_port_no_t port_no)`
Must be call after port PHY Reset (vtss_phy_reset).
- `vtss_rc vtss_phy_pre_system_reset (const vtss_inst_t inst, const vtss_port_no_t port_no)`
Must be call before a system reset.
- `vtss_rc vtss_phy_reset (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_reset_conf_t *const conf)`
Reset PHY.
- `vtss_rc vtss_phy_reset_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_reset_conf_t *conf)`
Get reset configuration.
- `vtss_rc vtss_phy_chip_temp_get (const vtss_inst_t inst, const vtss_port_no_t port_no, i16 *const temp)`
Get chip temperature.
- `vtss_rc vtss_phy_chip_temp_init (const vtss_inst_t inst, const vtss_port_no_t port_no)`
Init. chip temperature.
- `vtss_rc vtss_phy_conf_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_conf_t *const conf)`
Get PHY configuration.
- `vtss_rc vtss_phy_conf_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_conf_t *const conf)`
Set PHY configuration.
- `vtss_rc vtss_phy_status_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_port_status_t *const status)`
Get PHY status.
- `vtss_rc vtss_phy_cl37_lp_abil_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_port_status_t *const status)`
Get Clause37 Link pArtnr's ability.
- `vtss_rc vtss_phy_id_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_type_t *phy_id)`
Get the PHY type/id. The the Phy ID - can only be used after setting the operating mode.
- `vtss_rc vtss_phy_conf_1g_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_conf_1g_t *const conf)`
Get PHY 1G configuration.
- `vtss_rc vtss_phy_conf_1g_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_conf_1g_t *const conf)`
Set PHY 1G configuration.
- `vtss_rc vtss_phy_status_1g_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_status_1g_t *const status)`
Get PHY 1G status.
- `vtss_rc vtss_phy_power_conf_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_power_conf_t *const conf)`
Get PHY power configuration.
- `vtss_rc vtss_phy_power_conf_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_power_conf_t *const conf)`
Set PHY power configuration.
- `vtss_rc vtss_phy_power_status_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_power_status_t *const status)`
Get PHY power status.
- `vtss_rc vtss_phy_clock_conf_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_recov_clk_t clock_port, const vtss_phy_clock_conf_t *const conf)`
Set PHY clock configuration.

- `vtss_rc vtss_phy_clock_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_phy_recov_clk_t` clock_port, `vtss_phy_clock_conf_t` *const conf, `vtss_port_no_t` *const clock_source)

Get PHY clock configuration.

- `vtss_rc vtss_phy_i2c_read` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `u8` i2c_mux, const `u8` i2c_reg_addr, const `u8` i2c_device_addr, `u8` *const value, `u8` cnt, `BOOL` word_access)

I2C read.

- `vtss_rc vtss_phy_i2c_write` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `u8` i2c_mux, const `u8` i2c_reg_addr, const `u8` i2c_device_addr, `u8` *value, `u8` cnt, `BOOL` word_access)

I2C writes.

- `vtss_rc vtss_phy_read` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `u32` addr, `u16` *const value)

Read value from PHY register.

- `vtss_rc vtss_phy_read_page` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `u32` page, const `u32` addr, `u16` *const value)

Read value from PHY register at a specific page. Page register is set to standard page when read is done.

- `vtss_rc vtss_phy_mmd_read` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `u16` devad, const `u32` addr, `u16` *const value)

Read value from PHY mmd register.

- `vtss_rc vtss_phy_mmd_write` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `u16` devad, const `u32` addr, `u16` value)

Write value to PHY mmd register.

- `vtss_rc vtss_phy_write` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `u32` addr, const `u16` value)

Write value to PHY register.

- `vtss_rc vtss_phy_write_masked` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `u32` addr, const `u16` value, const `u16` mask)

Write masked value to PHY register.

- `vtss_rc vtss_phy_write_masked_page` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `u16` page, const `u16` addr, const `u16` value, const `u16` mask)

Write masked value to PHY register and setups the page register.

- `vtss_rc vtss_phy_gpio_mode` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `u8` gpio_no, const `vtss_phy_gpio_mode_t` mode)

Configure GPIO mode.

- `vtss_rc vtss_phy_gpio_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `u8` gpio_no, `BOOL` *value)

Get the value from a GPIO pin.

- `vtss_rc vtss_phy_gpio_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `u8` gpio_no, `BOOL` value)

Set the value of a GPIO pin.

- `vtss_rc vtss_phy_veriphy_start` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `u8` mode)

Start VeriPHY.

- `vtss_rc vtss_phy_veriphy_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_veriphy_result_t` *const result)

Get VeriPHY result.

- `vtss_rc vtss_phy_led_mode_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_phy_led_mode_select_t` led_mode_select)

Setting the LEDs blink mode.

- `vtss_rc vtss_phy_led_intensity_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_phy_led_intensity` intensity)

Setting the LEDs intensity.

- `vtss_rc vtss_phy_led_intensity_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_led_intensity` *intensity)

Getting the LEDs intensity.

- `vtss_rc vtss_phy_enhanced_led_control_init` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_phy_enhanced_led_control_t` conf)

Setting the enhanced LED control initial state (Should only be set once at startup)..
- `vtss_rc vtss_phy_enhanced_led_control_init_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_enhanced_led_control_t` *conf)

Getting the enhanced LED control initial state (Should only be set once at startup)..
- `vtss_rc vtss_phy_coma_mode_disable` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no)

Pulling the coma mode pin low.
- `vtss_rc vtss_phy_coma_mode_enable` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no)

Pulling the coma mode pin high (Set in coma mode if no external hardware is controlling the pin)
- `void vga_adc_debug` (const `vtss_inst_t` inst, `u8` vga_adc_pwr, `vtss_port_no_t` port_no)

debug function for Atom family Rev. A. chips
- `vtss_rc vtss_phy_port_eee_capable` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `BOOL` *eee_capable)

Get information about if a port is EEE capable.
- `vtss_rc vtss_phy_eee_ena` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `BOOL` enable)

Enabling / Disabling EEE (Energy Efficient Ethernet)
- `vtss_rc vtss_phy_eee_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_eee_conf_t` *conf)

Getting the current EEE (Energy Efficient Ethernet) configuration.
- `vtss_rc vtss_phy_eee_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_phy_eee_conf_t` conf)

Setting the EEE (Energy Efficient Ethernet) configuration.
- `vtss_rc vtss_phy_eee_power_save_state_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `BOOL` *rx_in_power_save_state, `BOOL` *tx_in_power_save_state)

Getting the if phy is currently powered save mode due to EEE.
- `vtss_rc vtss_phy_eee_link_partner_advertisements_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `u8` *advertisement)

Getting the EEE advertisement.
- `vtss_rc vtss_phy_event_enable_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_phy_event_t` ev_mask, const `BOOL` enable)

Enabling / Disabling of events.
- `vtss_rc vtss_phy_event_enable_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_event_t` *ev_mask)

Getting current interrupt event state.
- `vtss_rc vtss_phy_event_poll` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_event_t` *const ev_mask)

Polling for active events.
- `vtss_rc vtss_squelch_workaround` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `BOOL` enable)

Function for enabling/disabling squelch work around.
- `vtss_rc vtss_phy_csr_wr` (const `vtss_inst_t` inst, const `u16` page, const `vtss_port_no_t` port_no, const `u16` target, const `u32` csr_reg_addr, const `u32` value)

Function for writing to CSR registers.
- `vtss_rc vtss_phy_csr_rd` (const `vtss_inst_t` inst, const `u16` page, const `vtss_port_no_t` port_no, const `u16` target, const `u32` csr_reg_addr, `u32` *value)

Function for writing to CSR registers.
- `vtss_rc vtss_phy_statistic_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_statistic_t` *statistics)

debug function for getting phy statistics.
- `vtss_rc vtss_phy_do_page_chk_set` (const `vtss_inst_t` inst, const `BOOL` enable)

Debug function for enabling check of page register for all phy register accesses.
- `vtss_rc vtss_phy_do_page_chk_get` (const `vtss_inst_t` inst, `BOOL` *enable)

- Debug function for getting if check of page register is enabled.*

 - `vtss_rc vtss_phy_loopback_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_loopback_t` loopback)

Debug function for setting phy internal loopback.

 - `vtss_rc vtss_phy_loopback_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_loopback_t` *loopback)

Debug function for getting the current phy internal loopback.

 - `vtss_rc vtss_phy_is_8051_crc_ok` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `u16` code_length, `u16` expected_crc)

Debug function for checking if the phy firmware is loaded correctly.

 - `vtss_rc vtss_phy_cfg_ob_post0` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `u16` value)

Debug function for setting the ob post0 patch.

 - `vtss_rc vtss_phy_cfg_ib_cterm` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `u8` ib_cterm_ena, const `u8` ib_eq_mode)

Debug function for setting the ib cterm patch.

 - `vtss_rc vtss_phy_atom12_patch_settings_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `u8` *mcu_bus, `u8` *cfg_buf, `u8` *stat_buf)

Debug function for getting PHY setting set by the micro patches.

 - `void vtss_phy_reg_decode_status` (`vtss_port_no_t` port_no, `u16` lp_auto_neg_advertisement_reg, `u16` lp←1000base_t_status_reg, `u16` mii_status_reg, const `vtss_phy_conf_t` phy_setup, `vtss_port_status_t` *const status)

Function for updating the status via the result from PHY registers.

 - `vtss_rc vtss_phy_flowcontrol_decode_status` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `u16` lp←auto_neg_advertisement_reg, const `vtss_phy_conf_t` phy_setup, `vtss_port_status_t` *const status)

Function for finding flow control status based upon configuration and PHY registers.

 - `vtss_rc vtss_phy_debug_stat_print` (const `vtss_inst_t` inst, const `vtss_debug_printf_t` pr, const `vtss_port_no_t` port_no, const `BOOL` print_hdr)

debug function for printing PHY statistics

 - `vtss_rc vtss_phy_warm_start_failed_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no)

Function for checking if any issue were seen during warm-start.

 - `vtss_rc vtss_phy_debug_phyinfo_print` (const `vtss_inst_t` inst, const `vtss_debug_printf_t` pr, const `vtss_port_no_t` port_no, const `BOOL` print_hdr)

debug function for printing some of the internal PHY state/configurations

 - `vtss_rc vtss_phy_debug_register_dump` (const `vtss_inst_t` inst, const `vtss_debug_printf_t` pr, `BOOL` clear, const `vtss_port_no_t` port_no)

debug function for printing some of the internal PHY state/configurations

 - `vtss_rc vtss_phy_detect_base_ports` (const `vtss_inst_t` inst)

Function for making getting the API updated with base port number for all ports. MUST not be called unless all ports are detected.

 - `vtss_rc vtss_phy_ext_connector_loopback` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `BOOL` lpback)

Function for configuring External Connector Loopback.

 - `vtss_rc vtss_phy_serdes_sgmii_loopback` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `u16` mode)

Function for configuring MAC-SerDes(SGMII) Loopback.

 - `vtss_rc vtss_phy_serdes_fmedia_loopback` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `u16` mode)

Function for configuring Fibre-Media SerDes Loopback.

 - `vtss_rc vtss_phy_debug_regdump_print` (const `vtss_inst_t` inst, const `vtss_debug_printf_t` pr, const `vtss_port_no_t` port_no, const `vtss_port_no_t` page_no, const `BOOL` print_hdr)

debug function for printing some of the internal PHY Registers

 - `vtss_rc vtss_phy_wol_enable` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `BOOL` enable)

function to Enable or Disable WOL by enabling or disabling the interrupt

- `vtss_rc vtss_phy_wol_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_wol_conf_t` *const conf)

function to Get Wake-On-LAN configuration
- `vtss_rc vtss_phy_wol_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_phy_wol_conf_t` *const conf)

function to Set Wake-On-LAN configuration
- `vtss_rc vtss_phy_patch_settings_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `u8` *mcb_bus, `u8` *cfg_buf, `u8` *stat_buf)

Debug function for getting PHY setting set by the micro patches.
- `vtss_rc vtss_phy_serdes6g_rcpll_status_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_rcpll_status_t` *rcpll_status)

Debug function for getting PHY Serdes6G RC-PLL status.
- `vtss_rc vtss_phy_serdes1g_rcpll_status_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_rcpll_status_t` *rcpll_status)

Debug function for getting PHY Serdes1G RC-PLL status.
- `vtss_rc vtss_phy_lcpll_status_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_lcpll_status_t` *lcpll_status)

Debug function for getting PHY LC-PLL status.
- `vtss_rc vtss_phy_reset_lcpll` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no)

Debug function for Resetting the LCPLL for the PHY.
- `vtss_rc vtss_phy_sd6g_ob_post_rd` (`vtss_inst_t` inst, const `vtss_port_no_t` port_no, `u8` *ob_post0, `u8` *ob_post1)

Debug function for reading the 6G SerDes ob_post0 and ob_post1 values.
- `vtss_rc vtss_phy_sd6g_ob_post_wr` (`vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `u8` ob_post0, const `u8` ob_post1)

Debug function for modifying the 6G SerDes ob_post0 and ob_post1 values.
- `vtss_rc vtss_phy_sd6g_ob_lev_rd` (`vtss_inst_t` inst, const `vtss_port_no_t` port_no, `u8` *ob_level)

Debug function for reading the 6G SerDes ob_level value.
- `vtss_rc vtss_phy_sd6g_ob_lev_wr` (`vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `u8` ob_level)

Debug function for modifying the 6G SerDes ob_lev value.
- `vtss_rc vtss_phy_mac_media_inhibit_odd_start` (`vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `BOOL` mac_inhibit, const `BOOL` media_inhibit)

Function to modify the values for the MAC and MEDIA I/F Inhibit ODD Start Delay.
- `vtss_rc vtss_phy_fefi_get` (`vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_fefi_mode_t` *fefi)

Function to modify the values for the Far-End Fail Indication.
- `vtss_rc vtss_phy_fefi_set` (`vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_fefi_mode_t` fefi)

Function to modify the values for the Far-End Fail Indication.
- `vtss_rc vtss_phy_fefi_detect` (`vtss_inst_t` inst, const `vtss_port_no_t` port_no, `BOOL` *fefi_detect)

Function to get the status for the Far-End Fail Indication.
- `vtss_rc vtss_phy_mse_100m_get` (`vtss_inst_t` inst, const `vtss_port_no_t` port_no, `u32` *mse)

Function to get the Mean Squared Error (MSE), ie. Noise floor for 100m.
- `vtss_rc vtss_phy_mse_1000m_get` (`vtss_inst_t` inst, const `vtss_port_no_t` port_no, `u32` *mseA, `u32` *mseB, `u32` *mseC, `u32` *mseD)

Function to get the Mean Squared Error (MSE), ie. Noise floor for 1000m.
- `vtss_rc vtss_phy_read_tr_addr` (`vtss_inst_t` inst, const `vtss_port_no_t` port_no, `u16` tr_addr, `u16` *tr_lower, `u16` *tr_upper)

Debug Function to retrieve the values from Token Ring.
- `vtss_rc vtss_phy_is_viper_revB` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `BOOL` *is_viper_revB)

Polling for to determine if the Chip Type and revision is Viper Rev_B.
- `vtss_rc vtss_phy_ext_event_poll` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_event_t` *const ev_mask)

Polling for active EXT Interrupt events.

- `vtss_rc vtss_phy_status_inst_poll` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_port_status_t` *const status)
Get PHY status from the PHY Instance (Does not read PHY Registers).
- `vtss_rc vtss_phy_macsec_csr_sd6g_rd` (`vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `u16` target, const `u32` csr_reg_addr, `u32` *value)
Function for Reading from 6G serdes registers (applicable only on VIPER,ELISE)
- `vtss_rc vtss_phy_macsec_csr_sd6g_wr` (`vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `u16` target, const `u32` csr_reg_addr, `u32` value)
Function for Writing to 6G serdes registers (applicable only on VIPER,ELISE)

8.31.1 Detailed Description

PHY API.

This header file describes PHY control functions

8.31.2 Macro Definition Documentation

8.31.2.1 MAX_CFG_BUF_SIZE

```
#define MAX_CFG_BUF_SIZE 38
```

Defines the maximum size of the micro patch CFG buffer can be for all chip families.

Definition at line 154 of file vtss_phy_api.h.

8.31.2.2 MAX_STAT_BUF_SIZE

```
#define MAX_STAT_BUF_SIZE 8
```

Defines the number bytes in the PHY patch status array

Definition at line 155 of file vtss_phy_api.h.

8.31.2.3 VTSS_PHY_POWER_ACTIPHY_BIT

```
#define VTSS_PHY_POWER_ACTIPHY_BIT 0
```

Defines the bit used to signaling that ActiPhy is enabled

Definition at line 558 of file vtss_phy_api.h.

8.31.2.4 VTSS_PHY_POWER_DYNAMIC_BIT

```
#define VTSS_PHY_POWER_DYNAMIC_BIT 1
```

Defines the bit used to signaling that PerfectReach is enabled

Definition at line 559 of file vtss_phy_api.h.

8.31.2.5 VTSS_PHY_ACTIPHY_PWR

```
#define VTSS_PHY_ACTIPHY_PWR 100
```

ActiPHY power status

Definition at line 592 of file vtss_phy_api.h.

8.31.2.6 VTSS_PHY_LINK_DOWN_PWR

```
#define VTSS_PHY_LINK_DOWN_PWR 200
```

Link down power status

Definition at line 593 of file vtss_phy_api.h.

8.31.2.7 VTSS_PHY_LINK_UP_FULL_PWR

```
#define VTSS_PHY_LINK_UP_FULL_PWR 400
```

Link up full power status

Definition at line 594 of file vtss_phy_api.h.

8.31.2.8 VTSS_PHY_RECov_CLK1

```
#define VTSS_PHY_RECov_CLK1 0
```

PHY active clock out.

RCVRD_CLK1

Definition at line 617 of file vtss_phy_api.h.

8.31.2.9 VTSS_PHY_RECov_CLK2

```
#define VTSS_PHY_RECov_CLK2 1
```

RCVRD_CLK2

Definition at line 618 of file vtss_phy_api.h.

8.31.2.10 VTSS_PHY_RECov_CLK_NUM

```
#define VTSS_PHY_RECov_CLK_NUM 2
```

Number of recovered clocks

Definition at line 619 of file vtss_phy_api.h.

8.31.2.11 VTSS_PHY_PAGE_STANDARD

```
#define VTSS_PHY_PAGE_STANDARD 0x0000
```

Standard registers

Definition at line 740 of file vtss_phy_api.h.

8.31.2.12 VTSS_PHY_PAGE_EXTENDED

```
#define VTSS_PHY_PAGE_EXTENDED 0x0001
```

Extended registers

Definition at line 741 of file vtss_phy_api.h.

8.31.2.13 VTSS_PHY_PAGE_EXTENDED_2

```
#define VTSS_PHY_PAGE_EXTENDED_2 0x0002
```

Extended registers - page 2

Definition at line 742 of file vtss_phy_api.h.

8.31.2.14 VTSS_PHY_PAGE_EXTENDED_3

```
#define VTSS_PHY_PAGE_EXTENDED_3 0x0003
```

Extended registers - page 3

Definition at line 743 of file vtss_phy_api.h.

8.31.2.15 VTSS_PHY_PAGE_EXTENDED_4

```
#define VTSS_PHY_PAGE_EXTENDED_4 0x0004
```

Extended registers - page 4

Definition at line 744 of file vtss_phy_api.h.

8.31.2.16 VTSS_PHY_PAGE_GPIO

```
#define VTSS_PHY_PAGE_GPIO 0x0010
```

GPIO registers

Definition at line 745 of file vtss_phy_api.h.

8.31.2.17 VTSS_PHY_PAGE_1588

```
#define VTSS_PHY_PAGE_1588 0x1588
```

1588 (PTP) registers

Definition at line 746 of file vtss_phy_api.h.

8.31.2.18 VTSS_PHY_PAGE_MACSEC

```
#define VTSS_PHY_PAGE_MACSEC 0x0004
```

MACSEC page

Definition at line 747 of file vtss_phy_api.h.

8.31.2.19 VTSS_PHY_PAGE_TEST

```
#define VTSS_PHY_PAGE_TEST 0x2A30
```

Test registers

Definition at line 748 of file vtss_phy_api.h.

8.31.2.20 VTSS_PHY_PAGE_TR

```
#define VTSS_PHY_PAGE_TR 0x52B5
```

Token ring registers

Definition at line 749 of file vtss_phy_api.h.

8.31.2.21 VTSS_PHY_PAGE_0x2DAF

```
#define VTSS_PHY_PAGE_0x2DAF 0x2DAF
```

0x2DAF registers

Definition at line 750 of file vtss_phy_api.h.

8.31.2.22 VTSS_PHY_REG_STANDARD

```
#define VTSS_PHY_REG_STANDARD (VTSS_PHY_PAGE_STANDARD<<5)
```

Standard registers

Definition at line 753 of file vtss_phy_api.h.

8.31.2.23 VTSS_PHY_REG_EXTENDED

```
#define VTSS_PHY_REG_EXTENDED (VTSS_PHY_PAGE_EXTENDED<<5)
```

Extended registers

Definition at line 754 of file vtss_phy_api.h.

8.31.2.24 VTSS_PHY_REG_GPIO

```
#define VTSS_PHY_REG_GPIO (VTSS_PHY_PAGE_GPIO<<5)
```

GPIO registers

Definition at line 755 of file vtss_phy_api.h.

8.31.2.25 VTSS_PHY_REG_TEST

```
#define VTSS_PHY_REG_TEST (VTSS_PHY_PAGE_TEST<<5)
```

Test registers

Definition at line 756 of file vtss_phy_api.h.

8.31.2.26 VTSS_PHY_REG_TR

```
#define VTSS_PHY_REG_TR (VTSS_PHY_PAGE_TR<<5)
```

Token ring registers

Definition at line 757 of file vtss_phy_api.h.

8.31.2.27 VTSS_PHY_LINK_LOS_EV

```
#define VTSS_PHY_LINK_LOS_EV (1 << 0)
```

PHY link interrupt

Definition at line 1211 of file vtss_phy_api.h.

8.31.2.28 VTSS_PHY_LINK_FFAIL_EV

```
#define VTSS_PHY_LINK_FFAIL_EV (1 << 1)
```

PHY fast failure interrupt

Definition at line 1212 of file vtss_phy_api.h.

8.31.2.29 VTSS_PHY_LINK_AMS_EV

```
#define VTSS_PHY_LINK_AMS_EV (1 << 2)
```

PHY Automatic Media Sense

Definition at line 1213 of file vtss_phy_api.h.

8.31.2.30 VTSS_PHY_LINK_SPEED_STATE_CHANGE_EV

```
#define VTSS_PHY_LINK_SPEED_STATE_CHANGE_EV (1 << 3)
```

PHY link state change event

Definition at line 1214 of file vtss_phy_api.h.

8.31.2.31 VTSS_PHY_LINK_FDX_STATE_CHANGE_EV

```
#define VTSS_PHY_LINK_FDX_STATE_CHANGE_EV (1 << 4)
```

PHY FDX state change event

Definition at line 1215 of file vtss_phy_api.h.

8.31.2.32 VTSS_PHY_LINK_AUTO_NEG_ERROR_EV

```
#define VTSS_PHY_LINK_AUTO_NEG_ERROR_EV (1 << 5)
```

PHY Autonegotiation error event

Definition at line 1216 of file vtss_phy_api.h.

8.31.2.33 VTSS_PHY_LINK_AUTO_NEG_COMPLETE_EV

```
#define VTSS_PHY_LINK_AUTO_NEG_COMPLETE_EV (1 << 6)
```

PHY Autonegotiation complete event

Definition at line 1217 of file vtss_phy_api.h.

8.31.2.34 VTSS_PHY_LINK_INLINE_POW_DEV_DETECT_EV

```
#define VTSS_PHY_LINK_INLINE_POW_DEV_DETECT_EV (1 << 7)
```

PHY Inline powered device detect event

Definition at line 1218 of file vtss_phy_api.h.

8.31.2.35 VTSS_PHY_LINK_SYMBOL_ERR_INT_EV

```
#define VTSS_PHY_LINK_SYMBOL_ERR_INT_EV (1 << 8)
```

PHY Symbol error event

Definition at line 1219 of file vtss_phy_api.h.

8.31.2.36 VTSS_PHY_LINK_TX_FIFO_OVERFLOW_INT_EV

```
#define VTSS_PHY_LINK_TX_FIFO_OVERFLOW_INT_EV (1 << 9)
```

PHY TX fifo over/underflow detect event

Definition at line 1220 of file vtss_phy_api.h.

8.31.2.37 VTSS_PHY_LINK_RX_FIFO_OVERFLOW_INT_EV

```
#define VTSS_PHY_LINK_RX_FIFO_OVERFLOW_INT_EV (1 << 10)
```

PHY RX fifo over/underflow detect event

Definition at line 1221 of file vtss_phy_api.h.

8.31.2.38 VTSS_PHY_LINK_FALSE_CARRIER_INT_EV

```
#define VTSS_PHY_LINK_FALSE_CARRIER_INT_EV (1 << 11)
```

PHY false-carrier interrupt event

Definition at line 1222 of file vtss_phy_api.h.

8.31.2.39 VTSS_PHY_LINK_LINK_SPEED_DS_DETECT_EV

```
#define VTSS_PHY_LINK_LINK_SPEED_DS_DETECT_EV (1 << 12)
```

PHY Link speed downshift detect event

Definition at line 1223 of file vtss_phy_api.h.

8.31.2.40 VTSS_PHY_LINK_MASTER_SLAVE_RES_ERR_EV

```
#define VTSS_PHY_LINK_MASTER_SLAVE_RES_ERR_EV (1 << 13)
```

PHY master/slave resolution error event

Definition at line 1224 of file vtss_phy_api.h.

8.31.2.41 VTSS_PHY_LINK_RX_ER_INT_EV

```
#define VTSS_PHY_LINK_RX_ER_INT_EV (1 << 14)
```

PHY RX_ER interrupt event

Definition at line 1225 of file vtss_phy_api.h.

8.31.2.42 VTSS_PHY_LINK_EXTENDED_REG_INT_EV

```
#define VTSS_PHY_LINK_EXTENDED_REG_INT_EV (1 << 15)
```

PHY Use Extended Reg to Access interrupt event

Definition at line 1226 of file vtss_phy_api.h.

8.31.2.43 VTSS_PHY_LINK_WAKE_ON_LAN_INT_EV

```
#define VTSS_PHY_LINK_WAKE_ON_LAN_INT_EV (1 << 16)
```

PHY Wake-On-LAN interrupt event

Definition at line 1227 of file vtss_phy_api.h.

8.31.2.44 VTSS_PHY_LINK_EXT_EEE_WAKE_ERR_EV

```
#define VTSS_PHY_LINK_EXT_EEE_WAKE_ERR_EV (1 << 17)
```

PHY EEE Wake Error interrupt event

Definition at line 1229 of file vtss_phy_api.h.

8.31.2.45 VTSS_PHY_LINK_EXT_EEE_WAIT_TS_EV

```
#define VTSS_PHY_LINK_EXT_EEE_WAIT_TS_EV (1 << 18)
```

PHY EEE Wait Quit/Rx TS Timer interrupt event

Definition at line 1230 of file vtss_phy_api.h.

8.31.2.46 VTSS_PHY_LINK_EXT_EEE_WAIT_RX_TQ_EV

```
#define VTSS_PHY_LINK_EXT_EEE_WAIT_RX_TQ_EV (1 << 19)
```

PHY EEE Rx TQ Timer interrupt event

Definition at line 1231 of file vtss_phy_api.h.

8.31.2.47 VTSS_PHY_LINK_EXT_EEE_LINKFAIL_EV

```
#define VTSS_PHY_LINK_EXT_EEE_LINKFAIL_EV (1 << 20)
```

PHY EEE Link Fail interrupt event

Definition at line 1232 of file vtss_phy_api.h.

8.31.2.48 VTSS_PHY_LINK_EXT_RR_SW_COMPL_EV

```
#define VTSS_PHY_LINK_EXT_RR_SW_COMPL_EV (1 << 21)
```

PHY Ring Resiliency Switchover complete interrupt event

Definition at line 1233 of file vtss_phy_api.h.

8.31.2.49 VTSS_PHY_LINK_EXT_MACSEC_HOST_MAC_EV

```
#define VTSS_PHY_LINK_EXT_MACSEC_HOST_MAC_EV (1 << 22)
```

PHY MACSEC Host MAC interrupt event

Definition at line 1234 of file vtss_phy_api.h.

8.31.2.50 VTSS_PHY_LINK_EXT_MACSEC_LINE_MAC_EV

```
#define VTSS_PHY_LINK_EXT_MACSEC_LINE_MAC_EV (1 << 23)
```

PHY MACSEC Line MAC interrupt event

Definition at line 1235 of file vtss_phy_api.h.

8.31.2.51 VTSS_PHY_LINK_EXT_MACSEC_FC_BUFF_EV

```
#define VTSS_PHY_LINK_EXT_MACSEC_FC_BUFF_EV (1 << 24)
```

PHY MACSEC Flow Control Buff interrupt event

Definition at line 1236 of file vtss_phy_api.h.

8.31.2.52 VTSS_PHY_LINK_EXT_MACSEC_INGRESS_EV

```
#define VTSS_PHY_LINK_EXT_MACSEC_INGRESS_EV (1 << 25)
```

PHY MACSEC Ingress interrupt event

Definition at line 1237 of file vtss_phy_api.h.

8.31.2.53 VTSS_PHY_LINK_EXT_MACSEC_EGRESS_EV

```
#define VTSS_PHY_LINK_EXT_MACSEC_EGRESS_EV (1 << 26)
```

PHY MACSEC Egress interrupt event

Definition at line 1238 of file vtss_phy_api.h.

8.31.2.54 VTSS_PHY_LINK_EXT_MEM_INT_RING_EV

```
#define VTSS_PHY_LINK_EXT_MEM_INT_RING_EV (1 << 27)
```

PHY MEM Integrity Ring Control interrupt event

Definition at line 1239 of file vtss_phy_api.h.

8.31.2.55 MAX_WOL_MAC_ADDR_SIZE

```
#define MAX_WOL_MAC_ADDR_SIZE 6
```

Defines the maximum size WOL MAC ADDR, ie. 6 Octets.

Definition at line 1649 of file vtss_phy_api.h.

8.31.2.56 MAX_WOL_PASSWD_SIZE

```
#define MAX_WOL_PASSWD_SIZE 6
```

Defines the maximum size WOL Secure_On Password, ie. 6 Octets.

Definition at line 1650 of file vtss_phy_api.h.

8.31.2.57 MIN_WOL_PASSWD_SIZE

```
#define MIN_WOL_PASSWD_SIZE 4
```

Defines the minimum size WOL Secure_On Password, ie. 4 Octets.

Definition at line 1651 of file vtss_phy_api.h.

8.31.3 Typedef Documentation**8.31.3.1 vtss_phy_recov_clk_t**

```
typedef u16 vtss_phy_recov_clk_t
```

Container of recovered clock out identifier

Definition at line 620 of file vtss_phy_api.h.

8.31.3.2 vtss_phy_led_intensity

```
typedef u8 vtss_phy_led_intensity
```

PHY led intensity.

LED intensity from 0-200, LED intensity led_intensity * 0.5

Definition at line 1007 of file vtss_phy_api.h.

8.31.4 Enumeration Type Documentation

8.31.4.1 vtss_phy_led_mode_t

```
enum vtss_phy_led_mode_t
```

PHY LED modes.

Enumerator

LINK1000_ACTIVITY	No link in any speed on any media interface./Valid link at any speed on any media interface. Blink or pulse-stretch = Valid link at any speed on any media interface with activity present.
LINK100_ACTIVITY	No link in 1000BASE-T/Valid 1000BASE-T link. Blink or pulse-stretch = Valid 1000BASE-T link with activity present
LINK10_ACTIVITY	No link in 100BASE-T/Valid 100BASE-T link. Blink or pulse-stretch = Valid 100BASE-T link with activity present
LINK100_1000_ACTIVITY	No link in 10BASE-T/Valid 10BASE-T link. Blink or pulse-stretch = Valid 10BASE-T link with activity present
LINK10_1000_ACTIVITY	No link in 100BASE-T, 1000BASE-FX, or 1000BASE-TX/Valid 100BASE-T, 1000BASE-FX, or 1000BASE-TX link.Blink or pulse-stretch = Valid 100BASE-T, 1000BASE-FX, or 1000BASE-TX link with activity present.
LINK10_100_ACTIVITY	No link in 10BASE-T, 1000BASE-FX, or 1000BASE-TX/Valid 10BASE-T, 1000BASE-FX, or 1000BASE-TX link.Blink or pulse-stretch = Valid 10BASE-T, 1000BASE-FX, or 1000BASE-TX link with activity present.
LINK100BASE_FX_1000BASE_X_ACTIVITY	No link in 10BASE-T, 100BASE-FX, or 100BASE-TX/Valid 10BASE-T, 100BASE-FX, or 100BASE-TX link.Blink or pulse-stretch = Valid 10BASE-T, 100BASE-FX, or 100BASE-TX link with activity present.
DUPLEX_COLLISION	No link in 100BASE-FX or 1000BASE-X/ Valid 100BASE-FX or 1000BASE-X link. Blink or pulse-stretch = Valid 100BASE-FX or 1000BASE-X link with activity present.
COLLISION	Link established in half-duplex mode, or no link established. Link established in full-duplex mode.Blink or pulse-stretch = Link established in half-duplex mode but collisions are present

Enumerator

ACTIVITY	No collision detected.Blink or pulse-stretch = Collision detected.
BASE100_FX_1000BASE_X_FIBER_ACTIVITY	No activity present.Blink or pulse-stretch = Activity present
AUTONEGOTIATION_FAULT	No 100BASE-FX or 1000BASE-X activity present. Blink or pulse-stretch = 100BASE-FX or 1000BASE-X activity present
LINK1000BASE_X_ACTIVITY	No autonegotiation fault present., Autonegotiation fault occurred.
LINK100BASE_FX_ACTIVITY	No link in 1000BASE-X. Valid 1000BASE-X link.
BASE1000_ACTIVITY	No link in 100BASE-FX.
BASE100_FX_ACTIVITY	No 1000BASE-X activity present.Blink or pulse-stretch = 1000BASE-X activity present.
FORCE_LED_OFF	No 100BASE-FX activity present, Blink or pulse-stretch = 100BASE-FX activity present.
FORCE_LED_ON	De-asserts the LED
FAST_LINK_FAIL	Asserts the LED

Definition at line 102 of file vtss_phy_api.h.

8.31.4.2 vtss_phy_media_interface_t

```
enum vtss_phy_media_interface_t
```

PHY media interface type.

Enumerator

VTSS_PHY_MEDIA_IF_CU	Copper Interface
VTSS_PHY_MEDIA_IF_SFP_PASSTHRU	Fiber/Cu SFP Pass-thru
VTSS_PHY_MEDIA_IF_FI_1000BX	1000Base-X
VTSS_PHY_MEDIA_IF_FI_100FX	100Base-FX
VTSS_PHY_MEDIA_IF_AMS CU PASSTHRU	AMS - Cat5/SerDes/CuSFP passthru - Note the phy mode must be set to VTSS_PHY_MODE_ANEG
VTSS_PHY_MEDIA_IF_AMS FI PASSTHRU	AMS - Fiber passthru - Note the phy mode must be set to VTSS_PHY_MODE_ANEG
VTSS_PHY_MEDIA_IF_AMS CU_1000BX	AMS - Cat5/1000BX/CuSFP
VTSS_PHY_MEDIA_IF_AMS CU_100FX	AMS - Cat5/100FX/CuSFP

Definition at line 161 of file vtss_phy_api.h.

8.31.4.3 vtss_phy_mdi_t

```
enum vtss_phy_mdi_t
```

PHY media interface type.

Enumerator

VTSS_PHY_MDIX_AUTO	Copper media MDI auto detected
VTSS_PHY_MDI	Copper media forced to MDI
VTSS_PHY_MDIX	Copper media forced to MDI-X (Crossed cable)

Definition at line 175 of file vtss_phy_api.h.

8.31.4.4 rgmii_skew_delay_psec_t

```
enum rgmii_skew_delay_psec_t
```

RGMII skew values.

Enumerator

rgmii_skew_delay_200_psec	RGMII 200 Poco-Second Skew
rgmii_skew_delay_800_psec	RGMII 800 Poco-Second Skew
rgmii_skew_delay_1100_psec	RGMII 1100 Poco-Second Skew
rgmii_skew_delay_1700_psec	RGMII 1700 Poco-Second Skew
rgmii_skew_delay_2000_psec	RGMII 2000 Poco-Second Skew
rgmii_skew_delay_2300_psec	RGMII 2300 Poco-Second Skew
rgmii_skew_delay_2600_psec	RGMII 2600 Poco-Second Skew
rgmii_skew_delay_3400_psec	RGMII 3400 Poco-Second Skew

Definition at line 182 of file vtss_phy_api.h.

8.31.4.5 vtss_phy_forced_reset_t

```
enum vtss_phy_forced_reset_t
```

PHY forced reset interface type.

Enumerator

VTSS_PHY_FORCE_RESET	Default: Force reset of PHY, regardless of Config and HW MAC/MEDIA settings
VTSS_PHY_NOFORCE_RESET	Only reset PHY if SW Config and HW config of MAC/MEDIA settings differ

Definition at line 205 of file vtss_phy_api.h.

8.31.4.6 vtss_phy_pkt_mode_t

enum [vtss_phy_pkt_mode_t](#)

PHY packet mode configuration.

Enumerator

VTSS_PHY_PKT_MODE_IEEE_1_5_KB	IEEE NORMAL 1.5KB Pkt Length
VTSS_PHY_PKT_MODE_JUMBO_9_KB	JUMBO 9KB Pkt Length
VTSS_PHY_PKT_MODE_JUMBO_12_KB	JUMBO 12KB Pkt Length

Definition at line 211 of file vtss_phy_api.h.

8.31.4.7 vtss_phy_mode_t

enum [vtss_phy_mode_t](#)

PHY mode.

Enumerator

VTSS_PHY_MODE_ANEG	Auto negoatiation
VTSS_PHY_MODE_FORCED	Forced mode
VTSS_PHY_MODE_POWER_DOWN	Power down (disabled)

Definition at line 296 of file vtss_phy_api.h.

8.31.4.8 vtss_phy_fast_link_fail_t

enum [vtss_phy_fast_link_fail_t](#)

PHY fast link failure pin enable/disable.

Enumerator

VTSS_PHY_FAST_LINK_FAIL_ENABLE	Enable fast link failure pin
VTSS_PHY_FAST_LINK_FAIL_DISABLE	Disable fast link failure pin

Definition at line 325 of file vtss_phy_api.h.

8.31.4.9 `vtss_phy_sigdet_polarity_t`

enum `vtss_phy_sigdet_polarity_t`

PHY Sigdet pin polarity configuration.

Enumerator

<code>VTSS_PHY_SIGDET_POLARITY_ACT_LOW</code>	Set Sigdet polarity Active low
<code>VTSS_PHY_SIGDET_POLARITY_ACT_HIGH</code>	Set Sigdet polarity Active High

Definition at line 331 of file `vtss_phy_api.h`.

8.31.4.10 `vtss_phy_unidirectional_t`

enum `vtss_phy_unidirectional_t`

PHY Unidirectional enable/disable.

Enumerator

<code>VTSS_PHY_UNIDIRECTIONAL_DISABLE</code>	Disable Unidirectional (Default)
<code>VTSS_PHY_UNIDIRECTIONAL_ENABLE</code>	Enable Unidirectional

Definition at line 337 of file `vtss_phy_api.h`.

8.31.4.11 `vtss_phy_mac_serdes_pcs_sgmii_pre`

enum `vtss_phy_mac_serdes_pcs_sgmii_pre`

PHY MEDIA SerDes PCS Remote Fault Indication, See Clause 37, Table 37-3.

Enumerator

<code>VTSS_PHY_MAC_SERD_PCS_SGMII_IN_PRE_</code> ↔ NONE	MAC SerDes PCS Control, SGMII Input Preamble for 100BaseX - No Preamble Req'd
<code>VTSS_PHY_MAC_SERD_PCS_SGMII_IN_PRE_</code> ↔ ONE	MAC SerDes PCS Control, SGMII Input Preamble for 100BaseX - One-Byte Preamble Req'd
<code>VTSS_PHY_MAC_SERD_PCS_SGMII_IN_PRE_</code> ↔ TWO	MAC SerDes PCS Control, SGMII Input Preamble for 100BaseX - Two-Byte Preamble Req'd
<code>VTSS_PHY_MAC_SERD_PCS_SGMII_IN_PRE_</code> ↔ RSVD	MAC SerDes PCS Control, SGMII Input Preamble for 100BaseX - Reserved

Definition at line 343 of file `vtss_phy_api.h`.

8.31.4.12 vtss_phy_media_rem_fault_t

```
enum vtss_phy_media_rem_fault_t
```

PHY MEDIA SerDes PCS Remote Fault Indication, See Clause 37, Table 37-3.

Enumerator

VTSS_PHY_MEDIA_SERD_PCS_Rem_Fault_No_Error	Media SerDes PCS Control, Most Recent Clause 37 ANEG Exchg - Table 37-3
VTSS_PHY_MEDIA_SERD_PCS_Rem_Fault_Offline	Media SerDes PCS Control, Most Recent Clause 37 ANEG Exchg
VTSS_PHY_MEDIA_SERD_PCS_Rem_Fault_LINK_Fail	Media SerDes PCS Control, Most Recent Clause 37 ANEG Exchg
VTSS_PHY_MEDIA_SERD_PCS_Rem_Fault_Auto_Neg_Error	Media SerDes PCS Control, Most Recent Clause 37 ANEG Exchg

Definition at line 367 of file vtss_phy_api.h.

8.31.4.13 vtss_phy_media_force_ams_sel_t

```
enum vtss_phy_media_force_ams_sel_t
```

PHY AMS Force configuration.

Enumerator

VTSS_PHY_MEDIA_FORCE_AMS_Selection_Normal	Force AMS Override to Force Selection - Normal
VTSS_PHY_MEDIA_FORCE_AMS_Selection_Serdes	Force AMS Override to Force Selection - SerDes Media
VTSS_PHY_MEDIA_FORCE_AMS_Selection_Copper	Force AMS Override to Force Selection - Copper Media

Definition at line 388 of file vtss_phy_api.h.

8.31.4.14 vtss_phy_clk_source_t

```
enum vtss_phy_clk_source_t
```

PHY clock sources.

Enumerator

VTSS_PHY_CLK_Disabled	Recovered Clock Disable
VTSS_PHY_Serdes_Media	SerDes PHY
VTSS_PHY_Copper_Media	Copper PHY
VTSS_PHY_TCLK_Out	Transmitter TCLK
VTSS_PHY_Local_Xtal	Local XTAL

Definition at line 623 of file vtss_phy_api.h.

8.31.4.15 vtss_phy_freq_t

enum `vtss_phy_freq_t`

PHY clock frequencies.

Enumerator

<code>VTSS_PHY_FREQ_25M</code>	25 MHz
<code>VTSS_PHY_FREQ_125M</code>	125 MHz
<code>VTSS_PHY_FREQ_3125M</code>	31.25 MHz This is only valid on ATOM family - NOT Enzo

Definition at line 632 of file vtss_phy_api.h.

8.31.4.16 vtss_phy_clk_squelch

enum `vtss_phy_clk_squelch`

PHY clock squelch levels.

Enumerator

<code>VTSS_PHY_CLK_SQUELCH_MAX</code>	Automatically squelch clock to low when the link is not up, is unstable, is up in a mode that does not support the generation of arecovered clock (1000BASE-T master or 10BASE-T), or is up in EEE mode (100BASE-TX or 1000BASE-T slave).
<code>VTSS_PHY_CLK_SQUELCH_MED</code>	Same as <code>VTSS_PHY_CLK_SQUELCH_MAX</code> except that the clock is also generated in 1000BASE-T master and 10BASE-T link-up modes. This mode also generates a recovered clock output in EEE mode during reception of LP_IDLE.
<code>VTSS_PHY_CLK_SQUELCH_MIN</code>	Squelch only when the link is not up
<code>VTSS_PHY_CLK_SQUELCH_NONE</code>	Disable clock squelch.

Definition at line 639 of file vtss_phy_api.h.

8.31.4.17 vtss_phy_gpio_mode_t

enum `vtss_phy_gpio_mode_t`

GPIO pin operating mode.

Enumerator

VTSS_PHY_GPIO_ALT←_0	Set GPIO to as alternate function - e.g. SCL, SIGDET, 1588_SPI_CS or 1588_SPI_DO. Matches the alternate function "00" in the data-sheet
VTSS_PHY_GPIO_ALT←_1	Set GPIO to as alternate function 1 - Matches the alternate function "01" in the data-sheet
VTSS_PHY_GPIO_ALT←_2	Set GPIO to as alternate function 2 - Matches the alternate function "10" in the data-sheet
VTSS_PHY_GPIO_OUT	Set GPIO pin as output
VTSS_PHY_GPIO_IN	Set GPIO pin as input

Definition at line 885 of file vtss_phy_api.h.

8.31.4.18 lb_type

```
enum lb_type
```

Internal loop-back type.

Enumerator

VTSS_LB_1G_NONE	No looback
VTSS_LB_FAR_END	Loopback at far end (Loopback at cu side) - Only valid when Enable = TRUE
VTSS_LB_NEAR_END	Loopback at near end (Loopback at MAC side) - Only valid when Enable = TRUE

Definition at line 1377 of file vtss_phy_api.h.

8.31.4.19 vtss_wol_passwd_len_type_t

```
enum vtss_wol_passwd_len_type_t
```

Structure for Wake-On-LAN Password Length configuration.

Enumerator

VTSS_WOL_PASSWD_LEN←_4	PasswdLen=4 bytes
VTSS_WOL_PASSWD_LEN←_6	PasswdLen=6 bytes

Definition at line 1672 of file vtss_phy_api.h.

8.31.4.20 vtss_fefi_mode_t

enum `vtss_fefi_mode_t`

Far-End Failure Indication modes These settings map to PHY Reg23E3, bits 0 and 1.

Enumerator

<code>VTSS_100FX_FEFI_NORMAL</code>	Normal FEFI Operation, as specified by Reg23E3.1
<code>VTSS_100FX_FEFI_FORCE_SUPPRESS</code>	Force FEFI, as specified by Reg23E3.0
<code>VTSS_100FX_FEFI_FORCE_ENABLE</code>	Force FEFI, as specified by Reg23E3.0

Definition at line 1900 of file `vtss_phy_api.h`.

8.31.5 Function Documentation

8.31.5.1 vtss_phy_pre_reset()

```
vtss_rc vtss_phy_pre_reset (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no )
```

Must be call previous to port PHY Reset (`vtss_phy_reset`).

Parameters

<code>inst</code>	[IN] Target instance reference.
<code>port_no</code>	[IN] Port number (MUST be the first port for the chip).

Returns

Return code.

8.31.5.2 vtss_phy_post_reset()

```
vtss_rc vtss_phy_post_reset (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no )
```

Must be call after port PHY Reset (`vtss_phy_reset`).

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number (Any port with the chip can be used).

Returns

Return code.

8.31.5.3 vtss_phy_pre_system_reset()

```
vtss_rc vtss_phy_pre_system_reset (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no )
```

Must be call before a system reset.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number (Any port with the chip can be used).

Returns

Return code.

8.31.5.4 vtss_phy_reset()

```
vtss_rc vtss_phy_reset (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_reset_conf_t *const conf )
```

Reset PHY.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>conf</i>	[IN] Reset configuration.

Returns

Return code.

8.31.5.5 vtss_phy_reset_get()

```
vtss_rc vtss_phy_reset_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_reset_conf_t * conf )
```

Get reset configuration.

Parameters

<i>inst</i>	[IN] Target instance reference
<i>port_no</i>	[IN] Port number (Any port within the chip can be used)
<i>conf</i>	[OUT] Reset configuration

Returns

Return code.

8.31.5.6 vtss_phy_chip_temp_get()

```
vtss_rc vtss_phy_chip_temp_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    i16 *const temp )
```

Get chip temperature.

Parameters

<i>inst</i>	[IN] Target instance reference
<i>port_no</i>	[IN] Port number (Any port within the chip can be used).
<i>temp</i>	[OUT] Chip temperature

Returns

Return code.

8.31.5.7 vtss_phy_chip_temp_init()

```
vtss_rc vtss_phy_chip_temp_init (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no )
```

Init. chip temperature.

Parameters

<i>inst</i>	[IN] Target instance reference
<i>port_no</i>	[IN] Port number (Any port within the chip can be used).

Returns

Return code.

8.31.5.8 vtss_phy_conf_get()

```
vtss_rc vtss_phy_conf_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_conf_t *const conf )
```

Get PHY configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>conf</i>	[OUT] PHY configuration.

Returns

Return code.

8.31.5.9 vtss_phy_conf_set()

```
vtss_rc vtss_phy_conf_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_conf_t *const conf )
```

Set PHY configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>conf</i>	[IN] PHY configuration.

Returns

Return code.

8.31.5.10 vtss_phy_status_get()

```
vtss_rc vtss_phy_status_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_port_status_t *const status )
```

Get PHY status.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>status</i>	[OUT] PHY status.

Returns

Return code.

8.31.5.11 vtss_phy_cl37_lp_abil_get()

```
vtss_rc vtss_phy_cl37_lp_abil_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_port_status_t *const status )
```

Get Clause37 Link pArtner's ability.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>status</i>	[OUT] PHY status.

Returns

Return code.

8.31.5.12 vtss_phy_id_get()

```
vtss_rc vtss_phy_id_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_type_t * phy_id )
```

Get the PHY type/id. The the Phy ID - can only be used after setting the operating mode.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>phy_id</i>	[OUT] PHY Type/id.

Returns

Return code.

8.31.5.13 vtss_phy_conf_1g_get()

```
vtss_rc vtss_phy_conf_1g_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_conf_1g_t *const conf )
```

Get PHY 1G configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>conf</i>	[OUT] PHY 1G configuration.

Returns

Return code.

8.31.5.14 vtss_phy_conf_1g_set()

```
vtss_rc vtss_phy_conf_1g_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_conf_1g_t *const conf )
```

Set PHY 1G configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>conf</i>	[IN] PHY 1G configuration.

Returns

Return code.

8.31.5.15 vtss_phy_status_1g_get()

```
vtss_rc vtss_phy_status_1g_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_status_1g_t *const status )
```

Get PHY 1G status.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>status</i>	[OUT] PHY 1G status.

Returns

Return code.

8.31.5.16 vtss_phy_power_conf_get()

```
vtss_rc vtss_phy_power_conf_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_power_conf_t *const conf )
```

Get PHY power configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>conf</i>	[OUT] PHY power configuration.

Returns

Return code.

8.31.5.17 vtss_phy_power_conf_set()

```
vtss_rc vtss_phy_power_conf_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_power_conf_t *const conf )
```

Set PHY power configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>conf</i>	[IN] PHY power configuration.

Returns

Return code.

8.31.5.18 vtss_phy_power_status_get()

```
vtss_rc vtss_phy_power_status_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_power_status_t *const status )
```

Get PHY power status.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>status</i>	[OUT] PHY power configuration.

Returns

Return code.

8.31.5.19 vtss_phy_clock_conf_set()

```
vtss_rc vtss_phy_clock_conf_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_recov_clk_t clock_port,
    const vtss_phy_clock_conf_t *const conf )
```

Set PHY clock configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number to become clock source.
<i>clock_port</i>	[IN] Set configuration for this clock port.
<i>conf</i>	[IN] PHY clock configuration.

Returns

Return code.

8.31.5.20 vtss_phy_clock_conf_get()

```
vtss_rc vtss_phy_clock_conf_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_recov_clk_t clock_port,
    vtss_phy_clock_conf_t *const conf,
    vtss_port_no_t *const clock_source )
```

Get PHY clock configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number of the first port at this PHY instance.
<i>clock_port</i>	[IN] Get configuration for this clock port.
<i>conf</i>	[OUT] PHY clock configuration.
<i>clock_source</i>	[OUT] Port number that is clock source for this <i>clock_port</i> .

Returns

Return code.

8.31.5.21 vtss_phy_i2c_read()

```
vtss_rc vtss_phy_i2c_read (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const u8 i2c_mux,
    const u8 i2c_reg_addr,
    const u8 i2c_device_addr,
    u8 *const value,
    u8 cnt,
    BOOL word_access )
```

I2C read.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>i2c_mux</i>	[IN] The i2c clock mux
<i>i2c_reg_addr</i>	[IN] The i2c register address to access.
<i>i2c_device_addr</i>	[IN] The i2c address of the device to access.
<i>value</i>	[OUT] Pointer to where array which in going to contain the values read.
<i>cnt</i>	[IN] The number of registers to read. Note: The reg_addr is incremented by 1 for each of the read counts. If you want to read 16 bites registers (2 times 8 bits from the same register address), you need to do that by calling the vtss_phy_i2c_read twice, and not use the cnt (set cnt to 1).
<i>word_access</i>	[IN] Set to TRUE if the register data width is 16bit. FALSE = 8 bits data width

Returns

Return code.

8.31.5.22 vtss_phy_i2c_write()

```
vtss_rc vtss_phy_i2c_write (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const u8 i2c_mux,
    const u8 i2c_reg_addr,
    const u8 i2c_device_addr,
    u8 * value,
    u8 cnt,
    BOOL word_access )
```

I2C writes.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>i2c_mux</i>	[IN] The i2c clock mux

Parameters

<i>i2c_reg_addr</i>	[IN] The i2c register address to access.
<i>i2c_device_addr</i>	[IN] The i2c address of the device to access.
<i>value</i>	[IN] Pointer to where array containing the values to write.
<i>cnt</i>	[IN] The number of registers to write. Note: The reg_addr is incremented by 1 for each of the write counts. If you want to write 16 bites registers (2 times 8 bits to the same register address), you need to do that by calling the vtss_phy_i2c_write twice, and not use the cnt (set cnt to 1).
<i>word_access</i>	[IN] Set to TRUE if the register data width is 16bit. FALSE = 8 bits data width

Returns

Return code.

8.31.5.23 vtss_phy_read()

```
vtss_rc vtss_phy_read (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const u32 addr,
    u16 *const value )
```

Read value from PHY register.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>addr</i>	[IN] Register address. The page number is encoded in the 16 MSB.
<i>value</i>	[OUT] Register value.

Returns

Return code.

8.31.5.24 vtss_phy_read_page()

```
vtss_rc vtss_phy_read_page (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const u32 page,
    const u32 addr,
    u16 *const value )
```

Read value from PHY register at a specific page. Page register is set to standard page when read is done.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>page</i>	[IN] Page - Page do to the read at.
<i>addr</i>	[IN] Register address. The page number is encoded in the 16 MSB.
<i>value</i>	[OUT] Register value.

Returns

Return code.

8.31.5.25 vtss_phy_mmd_read()

```
vtss_rc vtss_phy_mmd_read (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const u16 devad,
    const u32 addr,
    u16 *const value )
```

Read value from PHY mmd register.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>devad</i>	[IN] Devad register address.
<i>addr</i>	[IN] Register address.
<i>value</i>	[OUT] Register value.

Returns

Return code.

8.31.5.26 vtss_phy_mmd_write()

```
vtss_rc vtss_phy_mmd_write (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const u16 devad,
    const u32 addr,
    u16 value )
```

Write value to PHY mmd register.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>devad</i>	[IN] Devad register address.
<i>addr</i>	[IN] Register address.
<i>value</i>	[OUT] Register value.

Returns

Return code.

8.31.5.27 vtss_phy_write()

```
vtss_rc vtss_phy_write (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const u32 addr,
    const u16 value )
```

Write value to PHY register.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>addr</i>	[IN] Register address. The page number is encoded in the 16 MSB.
<i>value</i>	[IN] Register value.

Returns

Return code.

8.31.5.28 vtss_phy_write_masked()

```
vtss_rc vtss_phy_write_masked (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const u32 addr,
    const u16 value,
    const u16 mask )
```

Write masked value to PHY register.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>addr</i>	[IN] Register address. The page number is encoded in the 16 MSB.
<i>value</i>	[IN] Register value.
<i>mask</i>	[IN] Register mask.

Returns

Return code.

8.31.5.29 vtss_phy_write_masked_page()

```
vtss_rc vtss_phy_write_masked_page (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const u16 page,
    const u16 addr,
    const u16 value,
    const u16 mask )
```

Write masked value to PHY register and setups the page register.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>page</i>	[IN] Page number.
<i>addr</i>	[IN] Register address. The page number is encoded in the 16 MSB.
<i>value</i>	[IN] Register value.
<i>mask</i>	[IN] Register mask.

Returns

Return code.

8.31.5.30 vtss_phy_gpio_mode()

```
vtss_rc vtss_phy_gpio_mode (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const u8 gpio_no,
    const vtss_phy_gpio_mode_t mode )
```

Configure GPIO mode.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Any port number within the chip containing the GPIO -
<i>gpio_no</i>	[IN] The GPIO number.
<i>mode</i>	[IN] The mode the GPIO pin should operate in.

Returns

VTSS_RC_OK when configuration was done correctly else error code.

8.31.5.31 vtss_phy_gpio_get()

```
vtss_rc vtss_phy_gpio_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const u8 gpio_no,
    BOOL * value )
```

Get the value from a GPIO pin.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Any port number within the chip containing the GPIO.
<i>gpio_no</i>	[IN] The GPIO number.
<i>value</i>	[OUT] Pointer to where to put the pin value (TRUE = pin input high, FALSE = pin input low)

Returns

VTSS_RC_OK if value is valid else error code.

8.31.5.32 vtss_phy_gpio_set()

```
vtss_rc vtss_phy_gpio_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const u8 gpio_no,
    BOOL value )
```

Set the value of a GPIO pin.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Any port number within the chip containing the GPIO.
<i>gpio_no</i>	[IN] The GPIO number.
<i>value</i>	[IN] The pin value. (TRUE = set pin high, FALSE = set pin low)

Returns

VTSS_RC_OK when setting was done correctly else error code.

8.31.5.33 vtss_phy_veriphy_start()

```
vtss_rc vtss_phy_veriphy_start (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const u8 mode )
```

Start VeriPHY.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>mode</i>	[IN] VeriPHY mode.

Returns

Return code.

8.31.5.34 vtss_phy_veriphy_get()

```
vtss_rc vtss_phy_veriphy_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_veriphy_result_t *const result )
```

Get VeriPHY result.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>result</i>	[OUT] VeriPHY result.

Returns

Return code.

8.31.5.35 vtss_phy_led_mode_set()

```
vtss_rc vtss_phy_led_mode_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_led_mode_select_t led_mode_select )
```

Setting the LEDs blink mode.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number the port in question.
<i>led_mode_select</i>	[IN] The LEDs mode

Returns

Return code.

8.31.5.36 vtss_phy_led_intensity_set()

```
vtss_rc vtss_phy_led_intensity_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_led_intensity intensity )
```

Setting the LEDs intensity.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number - Don't care for time being (Just set to a valid port), since all LEDs intensities are set to the same.
<i>intensity</i>	[IN] The LEDs intensities in % (0-100)

Returns

Return code.

8.31.5.37 vtss_phy_led_intensity_get()

```
vtss_rc vtss_phy_led_intensity_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_led_intensity * intensity )
```

Getting the LEDs intensity.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number - Don't care for time being (Just set to a valid port), since all LEDs intensities are set to the same.
<i>intensity</i>	[IN] The LEDs intensities in % (0-100)

Returns

Return code.

8.31.5.38 vtss_phy_enhanced_led_control_init()

```
vtss_rc vtss_phy_enhanced_led_control_init (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_enhanced_led_control_t conf )
```

Setting the enhanced LED control initial state (Should only be set once at startup)..

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>conf</i>	[IN] Enhanced LED control configuration

Returns

Return code.

8.31.5.39 vtss_phy_enhanced_led_control_init_get()

```
vtss_rc vtss_phy_enhanced_led_control_init_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_enhanced_led_control_t * conf )
```

Getting the enhanced LED control initial state (Should only be set once at startup)..

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>conf</i>	[IN] Enhanced LED control configuration

Returns

Return code.

8.31.5.40 vtss_phy_coma_mode_disable()

```
vtss_rc vtss_phy_coma_mode_disable (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no )
```

Pulling the coma mode pin low.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number (Any port number for the PHY which shall pull the coma mode pin low).

Returns

Return code.

8.31.5.41 vtss_phy_coma_mode_enable()

```
vtss_rc vtss_phy_coma_mode_enable (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no )
```

Pulling the coma mode pin high (Set in coma mode if no external hardware is controlling the pin)

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number (Any port number for the PHY which shall pull the coma mode pin low. remember to use same port number as when calling vtss_phy_coma_mode_disable).

Returns

Return code.

8.31.5.42 vga_adc_debug()

```
void vga_adc_debug (
    const vtss_inst_t inst,
```

```
    u8 vga_adc_pwr,
    vtss_port_no_t port_no )
```

debug function for Atom family Rev. A. chips

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>vga_adc_pwr</i>	[IN] allows VGA and/or ADC to power down for EEE
<i>port_no</i>	[IN] Port number (Any port number for the PHY which shall pull the coma mode pin low. remember to use same port number as when calling vtss_phy_coma_mode_disable).

8.31.5.43 vtss_phy_port_eee_capable()

```
vtss_rc vtss_phy_port_eee_capable (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    BOOL * eee_capable )
```

Get information about if a port is EEE capable.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>eee_capable</i>	[OUT] True if port is EEE capable else FALSE

Returns

Return code.

8.31.5.44 vtss_phy_eee_ena()

```
vtss_rc vtss_phy_eee_ena (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const BOOL enable )
```

Enabling / Disabling EEE (Energy Efficient Ethernet)

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>enable</i>	[IN] Enable EEE

Returns

Return code.

8.31.5.45 vtss_phy_eee_conf_get()

```
vtss_rc vtss_phy_eee_conf_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_eee_conf_t * conf )
```

Getting the current EEE (Energy Efficient Ethernet) configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>conf</i>	[OUT] EEE configuration.

Returns

Return code.

8.31.5.46 vtss_phy_eee_conf_set()

```
vtss_rc vtss_phy_eee_conf_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_eee_conf_t conf )
```

Setting the EEE (Energy Efficient Ethernet) configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>conf</i>	[OUT] EEE configuration.

Returns

Return code.

8.31.5.47 vtss_phy_eee_power_save_state_get()

```
vtss_rc vtss_phy_eee_power_save_state_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    BOOL * rx_in_power_save_state,
    BOOL * tx_in_power_save_state )
```

Getting the if phy is currently powered save mode due to EEE.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>rx_in_power_save_state</i>	[OUT] TRUE is phy rx part is in power save mode
<i>tx_in_power_save_state</i>	[OUT] TRUE is phy tx part is in power save mode

Returns

Return code.

8.31.5.48 vtss_phy_eee_link_partner_advertisements_get()

```
vtss_rc vtss_phy_eee_link_partner_advertisements_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    u8 * advertisement )
```

Getting the EEE advertisement.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>advertisement</i>	[OUT] Advertisement bit mask. Bit 0 = Link partner advertises 100BASE-T capability. Bit 1 = Link partner advertises 1000BASE-T capability.

Returns

Return code.

8.31.5.49 vtss_phy_event_enable_set()

```
vtss_rc vtss_phy_event_enable_set (
    const vtss_inst_t inst,
```

```
const vtss_port_no_t port_no,
const vtss_phy_event_t ev_mask,
const BOOL enable )
```

Enabling / Disabling of events.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>ev_mask</i>	[IN] Mask containing events that are enabled/disabled
<i>enable</i>	[IN] Enable/disable of event

Returns

Return code.

8.31.5.50 vtss_phy_event_enable_get()

```
vtss_rc vtss_phy_event_enable_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_event_t * ev_mask )
```

Getting current interrupt event state.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>ev_mask</i>	[IN] Mask containing events that are enabled/disabled

Returns

Return code.

8.31.5.51 vtss_phy_event_poll()

```
vtss_rc vtss_phy_event_poll (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_event_t *const ev_mask )
```

Polling for active events.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>ev_mask</i>	[OUT] Mask containing events that are active

Returns

Return code.

8.31.5.52 vtss_squelch_workaround()

```
vtss_rc vtss_squelch_workaround (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const BOOL enable )
```

Function for enabling/disabling squelch work around.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Any phy port with the chip
<i>enable</i>	[IN] TRUE = enable squelch workaround, FALSE = Disable squelch workaround

Returns

VTSS_RC_OK - Workaround was enabled/disable. VTSS_RC_ERROR - Squelch workaround patch not loaded

8.31.5.53 vtss_phy_csr_wr()

```
vtss_rc vtss_phy_csr_wr (
    const vtss_inst_t inst,
    const u16 page,
    const vtss_port_no_t port_no,
    const u16 target,
    const u32 csr_reg_addr,
    const u32 value )
```

Function for writing to CSR registers.

Parameters

<i>inst</i>	[IN] Target instance reference.
-------------	---------------------------------

Parameters

<i>port_no</i>	[IN] The port in question
<i>page</i>	[IN] The page for the CSR register access e.g. VTSS_PHY_PAGE_1588 or VTSS_PHY_PAGE_MACSEC
<i>target</i>	[IN] The CSR target
<i>csr_reg_addr</i>	[IN] The CSR register to write
<i>value</i>	[IN] The value to write

Returns

VTSS_RC_OK when new access can be done - VTSS_RC_ERROR if something went wrong and access was never granted.

8.31.5.54 vtss_phy_csr_rd()

```
vtss_rc vtss_phy_csr_rd (
    const vtss_inst_t inst,
    const u16 page,
    const vtss_port_no_t port_no,
    const u16 target,
    const u32 csr_reg_addr,
    u32 * value )
```

Function for writing to CSR registers.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Any phy port with the chip
<i>page</i>	[IN] The page for the CSR register access e.g. VTSS_PHY_PAGE_1588 or VTSS_PHY_PAGE_MACSEC
<i>target</i>	[IN] The CSR target
<i>csr_reg_addr</i>	[IN] The CSR register to read.
<i>value</i>	[IN] The value read

Returns

VTSS_RC_OK when new access can be done - VTSS_RC_ERROR if something went wrong and access was never granted.

8.31.5.55 vtss_phy_statistic_get()

```
vtss_rc vtss_phy_statistic_get (
    const vtss_inst_t inst,
```

```
const vtss_port_no_t port_no,
vtss_phy_statistic_t * statistics )
```

debug function for getting phy statistics.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Any phy port with the chip
<i>statistics</i>	[OUT] Pointer to where to put the statistics.

Returns

VTSS_RC_OK - Statistics is valid else statistics is invalid

8.31.5.56 vtss_phy_do_page_chk_set()

```
vtss_rc vtss_phy_do_page_chk_set (
    const vtss_inst_t inst,
    const BOOL enable )
```

Debug function for enabling check of page register for all phy register accesses.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>enable</i>	[IN] TRUE to enable phy page register check. FALSE to disable

Returns

Return code. VTSS_RC_OK if phy page check were set.

8.31.5.57 vtss_phy_do_page_chk_get()

```
vtss_rc vtss_phy_do_page_chk_get (
    const vtss_inst_t inst,
    BOOL * enable )
```

Debug function for getting if check of page register is enabled.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>enable</i>	[OUT] TRUE if phy page register check is enabled else FALSE

Returns

Return code. VTSS_RC_OK when enable is valid.

8.31.5.58 vtss_phy_loopback_set()

```
vtss_rc vtss_phy_loopback_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_loopback_t loopback )
```

Debug function for setting phy internal loopback.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Phy port that should have the internal loopback
<i>loopback</i>	[IN] Loopback type

Returns

Return code. VTSS_RC_OK if firmware is loaded correctly else VTSS_RC_ERROR

8.31.5.59 vtss_phy_loopback_get()

```
vtss_rc vtss_phy_loopback_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_loopback_t * loopback )
```

Debug function for getting the current phy internal loopback.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Phy port that with the internal loopback
<i>loopback</i>	[IN] Current loopback type

Returns

Return code. VTSS_RC_OK if firmware is loaded correctly else VTSS_RC_ERROR

8.31.5.60 vtss_phy_is_8051_crc_ok()

```
vtss_rc vtss_phy_is_8051_crc_ok (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    u16 code_length,
    u16 expected_crc )
```

Debug function for checking if the phy firmware is loaded correctly.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Must the first PHY port within the chip.
<i>code_length</i>	[IN] The length of the microcode patch
<i>expected_crc</i>	[IN] The expected CRC.

Returns

Return code. VTSS_RC_OK if firmware is loaded correctly else VTSS_RC_ERROR

8.31.5.61 vtss_phy_cfg_ob_post0()

```
vtss_rc vtss_phy_cfg_ob_post0 (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const u16 value )
```

Debug function for setting the ob post0 patch.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] PHY port within the chip.
<i>value</i>	[IN] The value to call the ob post0 patch with.

Returns

Return code. VTSS_RC_OK if patch were updated correct else VTSS_RC_ERROR

8.31.5.62 vtss_phy_cfg_ib_cterm()

```
vtss_rc vtss_phy_cfg_ib_cterm (
    const vtss_inst_t inst,
```

```
const vtss_port_no_t port_no,
const u8 ib_cterm_ena,
const u8 ib_eq_mode )
```

Debug function for setting the ib cterm patch.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] PHY port within the chip.
<i>ib_cterm_ena</i>	[IN] The value of ib_cterm_ena to call the ib cterm patch with.
<i>ib_eq_mode</i>	[IN] The value of ib_eq_mode to call the ib cterm patch with.

Returns

Return code. VTSS_RC_OK if patch were updated correct else VTSS_RC_ERROR

8.31.5.63 vtss_phy_atom12_patch_settings_get()

```
vtss_rc vtss_phy_atom12_patch_settings_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    u8 * mcb_bus,
    u8 * cfg_buf,
    u8 * stat_buf )
```

Debug function for getting PHY setting set by the micro patches.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] PHY port within the chip.
<i>mcb_bus</i>	[INOUT] If set to 2 the returned data is for LCPLL/RComp else the mcb_bus is returning 0 if the data is for 1G, and returning 1 if the data is for 6G .
<i>cfg_buf</i>	[IN] Pointer to array where to put the current settings.
<i>stat_buf</i>	[IN] Pointer to array where to put the current status.

Returns

Return code. VTSS_RC_OK if patch were updated correct else VTSS_RC_ERROR

8.31.5.64 vtss_phy_reg_decode_status()

```
void vtss_phy_reg_decode_status (
    vtss_port_no_t port_no,
```

```

    u16 lp_auto_neg_advertisement_reg,
    u16 lp_1000base_t_status_reg,
    u16 mii_status_reg,
    const vtss_phy_conf_t phy_setup,
    vtss_port_status_t *const status )

```

Function for updating the status via the result from PHY registers.

Parameters

<i>port_no</i>	[IN] The port in question.
<i>lp_auto_neg_advertisement_reg</i>	[IN] The value from the register containing the Link partners auto negotiation advertisement (Standard page 5)
<i>lp_1000base_t_status_reg</i>	[IN] The value from the register containing the Link partners 1000BASE-T Status (Standard page 10)
<i>mii_status_reg</i>	[IN] The value from the register containing mii status (Standard page 1)
<i>phy_setup</i>	[IN] The phy configuration setup
<i>status</i>	[INOUT] Pointer to where to put the result

8.31.5.65 vtss_phy_flowcontrol_decode_status()

```

vtss_rc vtss_phy_flowcontrol_decode_status (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    u16 lp_auto_neg_advertisement_reg,
    const vtss_phy_conf_t phy_setup,
    vtss_port_status_t *const status )

```

Function for finding flow control status based upon configuration and PHY registers.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] The port in question.
<i>lp_auto_neg_advertisement_reg</i>	[IN] The value from the register containing the Link partners auto negotiation advertisement (Standard page 5)
<i>phy_setup</i>	[IN] The phy configuration setup
<i>status</i>	[INOUT] Pointer to where to put the result *

Returns

Return code. VTSS_RC_OK if not errors were seen during warm-start else VTSS_RC_ERROR.

8.31.5.66 vtss_phy_debug_stat_print()

```

vtss_rc vtss_phy_debug_stat_print (
    const vtss_inst_t inst,

```

```
const vtss_debug_printf_t pr,
const vtss_port_no_t port_no,
const BOOL print_hdr )
```

debug function for printing PHY statistics

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>pr</i>	[IN] Function pointer to print function e.g. CPRINTF
<i>port_no</i>	[IN] Port in question
<i>print_hdr</i>	[IN] Set to TRUE to print header and counters. Set FALSE to only print counters

Returns

Return code. VTSS_RC_OK if not errors were seen during warm-start else VTSS_RC_ERROR.

8.31.5.67 vtss_phy_warm_start_failed_get()

```
vtss_rc vtss_phy_warm_start_failed_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no )
```

Function for checking if any issue were seen during warm-start.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] The port in question.

Returns

Return code. VTSS_RC_OK if not errors were seen during warm-start else VTSS_RC_ERROR.

8.31.5.68 vtss_phy_debug_phyinfo_print()

```
vtss_rc vtss_phy_debug_phyinfo_print (
    const vtss_inst_t inst,
    const vtss_debug_printf_t pr,
    const vtss_port_no_t port_no,
    const BOOL print_hdr )
```

debug function for printing some of the internal PHY state/configurations

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>pr</i>	[IN] Function pointer to print function e.g. CPRINTF
<i>port_no</i>	[IN] Port in question
<i>print_hdr</i>	[IN] Set to TRUE to print header and phy state info. Set FALSE to only print state info. Normally this is set to TRUE for the first port called, and FALSE for the rest in order to only get one (common) header

Returns

Return code. VTSS_RC_OK if not errors were seen during warm-start else VTSS_RC_ERROR.

8.31.5.69 vtss_phy_debug_register_dump()

```
vtss_rc vtss_phy_debug_register_dump (
    const vtss_inst_t inst,
    const vtss_debug_printf_t pr,
    BOOL clear,
    const vtss_port_no_t port_no )
```

debug function for printing some of the internal PHY state/configurations

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>pr</i>	[IN] Function pointer to print function e.g. CPRINTF
<i>clear</i>	[IN] Set to TRUE to clear the counters & Stickt bits if any
<i>port_no</i>	[IN] Port in question

Returns

Return code. VTSS_RC_OK if not errors were seen during warm-start else VTSS_RC_ERROR.

8.31.5.70 vtss_phy_detect_base_ports()

```
vtss_rc vtss_phy_detect_base_ports (
    const vtss_inst_t inst )
```

Function for making getting the API updated with base port number for all ports. MUST not be called unless all ports are detected.

Parameters

<i>inst</i>	[IN] Target instance reference.
-------------	---------------------------------

Returns

Return code. VTSS_RC_OK if all base ports were updated correctly else error code.

8.31.5.71 vtss_phy_ext_connector_loopback()

```
vtss_rc vtss_phy_ext_connector_loopback (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const BOOL lpback )
```

Function for configuring External Connector Loopback.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] The port in question.
<i>lpback</i>	[IN] TRUE=Loopback ON, FALSE=Loopback OFF

Returns

Return code. VTSS_RC_OK if no errors were seen during warm-start else VTSS_RC_ERROR.

8.31.5.72 vtss_phy_serdes_sgmii_loopback()

```
vtss_rc vtss_phy_serdes_sgmii_loopback (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const u16 mode )
```

Function for configuring MAC-SerDes(SGMII) Loopback.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] The port in question.
<i>mode</i>	[IN] serdes mode and port mode: bits 15:12: 0x8= Fibre Media or 0x9=SGMII/QSGMII MAC mode: bits 11:8: Port Address (0-3) mode: bits 7:4: Loopback Type: 0=No,1=Pad,2=Input,4=Facility,8=Equipment mode: bits 3:0: 0x2

Returns

Return code. VTSS_RC_OK if no errors were seen during warm-start else VTSS_RC_ERROR.

8.31.5.73 vtss_phy_serdes_fmedia_loopback()

```
vtss_rc vtss_phy_serdes_fmedia_loopback (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const u16 mode )
```

Function for configuring Fibre-Media SerDes Loopback.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] The port in question.
<i>mode</i>	[IN] serdes mode and port mode: bits 15:12: 0x8= Fibre Media or 0x9=SGMII/QSGMII MAC mode: bits 11:8: Port Address (0-3) mode: bits 7:4: Loopback Type: 0=No,1=Pad,2=Input,4=Facility,8=Equipment mode: bits 3:0: 0x2

Returns

Return code. VTSS_RC_OK if not errors were seen during warm-start else VTSS_RC_ERROR.

8.31.5.74 vtss_phy_debug_regdump_print()

```
vtss_rc vtss_phy_debug_regdump_print (
    const vtss_inst_t inst,
    const vtss_debug_printf_t pr,
    const vtss_port_no_t port_no,
    const vtss_port_no_t page_no,
    const BOOL print_hdr )
```

debug function for printing some of the internal PHY Registers

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>pr</i>	[IN] Function pointer to print function e.g. CPRINTF
<i>port_no</i>	[IN] Port in question
<i>page_no</i>	[IN] Page No in question, 0, 1, 2, 3, 1588, MACSEC, TEST, TR
<i>print_hdr</i>	[IN] Set to TRUE to print header and phy state info. Set FALSE to only print state info. Normally this is set to TRUE for the first port called, and FALSE for the rest in order to only get one (common) header

Returns

Return code. VTSS_RC_OK if not errors were seen during warm-start else VTSS_RC_ERROR.

8.31.5.75 vtss_phy_wol_enable()

```
vtss_rc vtss_phy_wol_enable (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const BOOL enable )
```

function to Enable or Disable WOL by enabling or disabling the interrupt

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port in question
<i>enable</i>	[IN] Boolean, Enable=TRUE or 1, Disable=False or 0

Returns

Return code. VTSS_RC_OK if no errors.

8.31.5.76 vtss_phy_wol_conf_get()

```
vtss_rc vtss_phy_wol_conf_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_wol_conf_t *const conf )
```

function to Get Wake-On-LAN configuration

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port in question
<i>conf</i>	[IN] Ptr to WoL Structure <code>vtss_phy_wol_conf_t</code> to be filled out by API

Returns

Return code. VTSS_RC_OK if no errors.

8.31.5.77 vtss_phy_wol_conf_set()

```
vtss_rc vtss_phy_wol_conf_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_wol_conf_t *const conf )
```

function to Set Wake-On-LAN configuration

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port in question
<i>conf</i>	[IN] Ptr to WoL Structure vtss_phy_wol_conf_t filled out by User

Returns

Return code. VTSS_RC_OK if no errors.

8.31.5.78 vtss_phy_patch_settings_get()

```
vtss_rc vtss_phy_patch_settings_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    u8 * mcb_bus,
    u8 * cfg_buf,
    u8 * stat_buf )
```

Debug function for getting PHY setting set by the micro patches.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] PHY port within the chip.
<i>mcb_bus</i>	[INOUT] If set to 2 the returned data is for LCPLL/RComp else the mcb_bus is returning 0 if the data is for 1G, and returning 1 if the data is for 6G .
<i>cfg_buf</i>	[IN] Pointer to array where to put the current settings.
<i>stat_buf</i>	[IN] Pointer to array where to put the current status.

Returns

Return code. VTSS_RC_OK if patch were updated correct else VTSS_RC_ERROR

8.31.5.79 vtss_phy_serdes6g_rcpll_status_get()

```
vtss_rc vtss_phy_serdes6g_rcpll_status_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_rcpll_status_t * rcpll_status )
```

Debug function for getting PHY Serdes6G RC-PLL status.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] PHY port within the chip.
<i>rcpll_status</i>	[OUT] Pointer to RC-PLL structure vtss_rcpll_status_t to get the status.

Returns

Return code. VTSS_RC_OK if patch were updated correct else VTSS_RC_ERROR

8.31.5.80 vtss_phy_serdes1g_rcpll_status_get()

```
vtss_rc vtss_phy_serdes1g_rcpll_status_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_rcpll_status_t * rcpll_status )
```

Debug function for getting PHY Serdes1G RC-PLL status.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] PHY port within the chip.
<i>rcpll_status</i>	[OUT] Pointer to RC-PLL structure vtss_rcpll_status_t to get the status.

Returns

Return code. VTSS_RC_OK if patch were updated correct else VTSS_RC_ERROR

8.31.5.81 vtss_phy_lcpll_status_get()

```
vtss_rc vtss_phy_lcpll_status_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_lcpll_status_t * lcpll_status )
```

Debug function for getting PHY LC-PLL status.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] PHY port within the chip.
<i>lcpll_status</i>	[OUT] Pointer to LC-PLL structure vtss_lcpll_status_t to get the status.

Returns

Return code. VTSS_RC_OK if patch were updated correct else VTSS_RC_ERROR

8.31.5.82 vtss_phy_reset_lcpll()

```
vtss_rc vtss_phy_reset_lcpll (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no )
```

Debug function for Resetting the LCPLL for the PHY.

Note

: This would occur PRIOR to calling PHY pre-reset(), reset(), and post-reset() functions, immediately after HW Reset At this point in the process, the PHY API does not know the PHY Base Port Number. If the Calling application uses the Base Port number, the LCPLL is reset and VTSS_RC_OK is returned If the Calling application uses any other port number, VTSS_RC_ERR_PHY_BASE_NO_NOT_FOUND is returned and no action is taken

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] PHY port within the chip.

Returns

Return code. VTSS_RC_OK if LCPLL reset correctly VTSS_RC_ERR_PHY_BASE_NO_NOT_FOUND if the port_no used was not the base_port_no of the PHY, ie. No action taken VTSS_RC_ERROR if and error occurred.

8.31.5.83 vtss_phy_sd6g_ob_post_rd()

```
vtss_rc vtss_phy_sd6g_ob_post_rd (
    vtss_inst_t inst,
    const vtss_port_no_t port_no,
    u8 * ob_post0,
    u8 * ob_post1 )
```

Debug function for reading the 6G SerDes ob_post0 and ob_post1 values.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] PHY port within the chip.
<i>ob_post0</i>	[OUT] ob_post0 settings in serdes6g_ob_cfg for 6G SerDes Macro (See TN1052), 6 bits, default:0, Customizable.
<i>ob_post1</i>	[OUT] ob_post1 settings in serdes6g_ob_cfg for 6G SerDes Macro (See TN1052), 5 bits, default:0, Do not change.

Returns

Return code. VTSS_RC_OK if all Ok VTSS_RC_ERROR if and error occurred.

8.31.5.84 vtss_phy_sd6g_ob_post_wr()

```
vtss_rc vtss_phy_sd6g_ob_post_wr (
    vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const u8 ob_post0,
    const u8 ob_post1 )
```

Debug function for modifying the 6G SerDes ob_post0 and ob_post1 values.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] PHY port within the chip.
<i>ob_post0</i>	[IN] Modify ob_post0 settings in serdes6g_ob_cfg for 6G SerDes Macro (See TN1052), 6 bits, default:0, Customizeable.
<i>ob_post1</i>	[IN] Modify ob_post1 settings in serdes6g_ob_cfg for 6G SerDes Macro (See TN1052), 5 bits, default:0, Do not change.

Returns

Return code. VTSS_RC_OK if all Ok VTSS_RC_ERROR if and error occurred.

8.31.5.85 vtss_phy_sd6g_ob_lev_rd()

```
vtss_rc vtss_phy_sd6g_ob_lev_rd (
    vtss_inst_t inst,
    const vtss_port_no_t port_no,
    u8 * ob_level )
```

Debug function for reading the 6G SerDes ob_level value.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] PHY port within the chip.
<i>ob_level</i>	[OUT] ob_level settings in serdes6g_ob_cfg1 for 6G SerDes Macro (See TN1052), 6 bits, default:0x18, Customizeable Amplitude Control.

Returns

Return code. VTSS_RC_OK if all Ok VTSS_RC_ERROR if and error occurred.

8.31.5.86 vtss_phy_sd6g_ob_lev_wr()

```
vtss_rc vtss_phy_sd6g_ob_lev_wr (
    vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const u8 ob_level )
```

Debug function for modifying the 6G SerDes ob_lev value.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] PHY port within the chip.
<i>ob_level</i>	[IN] Modify ob_lev settings in serdes6g_ob_cfg1 for 6G SerDes Macro (See TN1052), 6 bits, default:0x18, Customizable Amplitude Control.

Returns

Return code. VTSS_RC_OK if all Ok VTSS_RC_ERROR if and error occurred.

8.31.5.87 vtss_phy_mac_media_inhibit_odd_start()

```
vtss_rc vtss_phy_mac_media_inhibit_odd_start (
    vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const BOOL mac_inhibit,
    const BOOL media_inhibit )
```

Function to modify the values for the MAC and MEDIA I/F Inhibit ODD Start Delay.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] PHY port within the chip.
<i>mac_inhibit</i>	[IN] Modify Inhibit MAC Odd-start Delay settings, ie. Reg 16E3.2.
<i>media_inhibit</i>	[IN] Modify Inhibit MEDIA Odd-start Delay settings, ie. Reg 23E3.4.

Returns

Return code. VTSS_RC_OK if all Ok VTSS_RC_ERROR if and error occurred.

8.31.5.88 vtss_phy_fefi_get()

```
vtss_rc vtss_phy_fefi_get (
    vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_fefi_mode_t * fefi )
```

Function to modify the values for the Far-End Fail Indication.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] PHY port within the chip.
<i>fefi</i>	[OUT] PHY port Far End Failure Indicator Config.

Returns

Return code. VTSS_RC_OK if all Ok VTSS_RC_ERROR if and error occurred.

8.31.5.89 vtss_phy_fefi_set()

```
vtss_rc vtss_phy_fefi_set (
    vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_fefi_mode_t fefi )
```

Function to modify the values for the Far-End Fail Indication.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] PHY port within the chip.
<i>fefi</i>	[IN] PHY port Far End Failure Indicator Config.

Returns

Return code. VTSS_RC_OK if all Ok VTSS_RC_ERROR if and error occurred.

8.31.5.90 vtss_phy_fefi_detect()

```
vtss_rc vtss_phy_fefi_detect (
    vtss_inst_t inst,
    const vtss_port_no_t port_no,
    BOOL * fefi_detect )
```

Function to get the status for the Far-End Fail Indication.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] PHY port within the chip.
<i>fefi_detect</i>	[OUT] PHY port Far End Failure Indicator

Returns

Return code. VTSS_RC_OK if all Ok VTSS_RC_ERROR if and error occurred.

8.31.5.91 vtss_phy_mse_100m_get()

```
vtss_rc vtss_phy_mse_100m_get (
    vtss_inst_t inst,
    const vtss_port_no_t port_no,
    u32 * mse )
```

Function to get the Mean Squared Error (MSE), ie. Noise floor for 100m.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] PHY port within the chip.
<i>mse</i>	[OUT] PHY port MSE Value In order to convert the returned mse value to dB, do the following This computation would be required for each channel pair $mse_dbl = mse / (1024 * 2048)$; Note: Convert to Double and Scale by $2^{21} = 1024 * 2048$ $mse_dbl = 20 * \log10(mse_dbl)$; Note: Convert to dB Nominal Computed Values for mse_dbl are: -22dB to -31dB

Returns

Return code. VTSS_RC_OK if all Ok VTSS_RC_ERROR if and error occurred.

8.31.5.92 vtss_phy_mse_1000m_get()

```
vtss_rc vtss_phy_mse_1000m_get (
    vtss_inst_t inst,
    const vtss_port_no_t port_no,
    u32 * mseA,
    u32 * mseB,
    u32 * mseC,
    u32 * mseD )
```

Function to get the Mean Squared Error (MSE), ie. Noise floor for 1000m.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] PHY port within the chip.
<i>mseA</i>	[OUT] PHY port MSE Chan A Value
<i>mseB</i>	[OUT] PHY port MSE Chan B Value
<i>mseC</i>	[OUT] PHY port MSE Chan C Value
<i>mseD</i>	[OUT] PHY port MSE Chan D Value In order to convert the returned mse value to dB, do the following This computation would be required for each channel pair $mse_dbl = mse / (1024 * 2048)$; Note: Convert to Double and Scale by $2^{21} = 1024 * 2048$ $mse_dbl = 20 * \log_{10}(mse_dbl)$; Note: Convert to dB Nominal Computed Values for mse_dbl are: -22dB to -31dB

Returns

Return code. VTSS_RC_OK if all Ok VTSS_RC_ERROR if and error occurred.

8.31.5.93 vtss_phy_read_tr_addr()

```
vtss_rc vtss_phy_read_tr_addr (
    vtss_inst_t inst,
    const vtss_port_no_t port_no,
    u16 tr_addr,
    u16 * tr_lower,
    u16 * tr_upper )
```

Debug Function to retrieve the values from Token Ring.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] PHY port within the chip.
<i>tr_addr</i>	[IN] Token Ring ADDR (TR16) to Read
<i>tr_lower</i>	[OUT] Token Ring Lower 16bits of Value (TR17)
<i>tr_upper</i>	[OUT] Token Ring Upper 16bits of Value (TR18)

Returns

Return code. VTSS_RC_OK if all Ok VTSS_RC_ERROR if and error occurred.

8.31.5.94 vtss_phy_is_viper_revB()

```
vtss_rc vtss_phy_is_viper_revB (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    BOOL * is_viper_revB )
```

Polling for to determine if the Chip Type and revision is Viper Rev_B.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>is_viper_revB</i>	[OUT] Boolean to indicate that the Chip/Rev is Viper RevB

Returns

Return code.

8.31.5.95 vtss_phy_ext_event_poll()

```
vtss_rc vtss_phy_ext_event_poll (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_event_t *const ev_mask )
```

Polling for active EXT Interrupt events.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>ev_mask</i>	[OUT] Mask containing events that are active

Returns

Return code.

NOTE: Viper Rev. B Self-Clearing Interrupt Stuck ON Work-Around Normally, the API function: [vtss_phy_event_poll\(\)](#) handles ALL Interrupts. This API is a work-around for Viper family (VSC8584/VSC8582/VSC8575/VSC8564/↔ VSC8562/VSC8586) Viper Rev_B has a Bug which prevents EXT INT (Reg26.5) and AMS INT (Reg26.4) from Clearing properly (MDINT stays asserted), This results in MDINT Stuck ON if one of these INT's are ever triggered, putting the system into a Stuck Interrupt situation This API can be used to directly Poll for the events in Extended Interrupt Status Reg. 29E2.

8.31.5.96 vtss_phy_status_inst_poll()

```
vtss_rc vtss_phy_status_inst_poll (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_port_status_t *const status )
```

Get PHY status from the PHY Instance (Does not read PHY Registers).

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>status</i>	[OUT] PHY status.

Returns

Return code.

8.31.5.97 vtss_phy_macsec_csr_sd6g_rd()

```
vtss_rc vtss_phy_macsec_csr_sd6g_rd (
    vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const u16 target,
    const u32 csr_reg_addr,
    u32 * value )
```

Function for Reading from 6G serdes registers (applicable only on VIPER,ELISE)

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Any phy port with the chip
<i>target</i>	[IN] The CSR target
<i>csr_reg_addr</i>	[IN] The CSR register to read,
<i>value</i>	[OUT] The value read

Returns

VTSS_RC_OK when new access can be done - VTSS_RC_ERROR if something went wrong and access was never granted.

8.31.5.98 vtss_phy_macsec_csr_sd6g_wr()

```
vtss_rc vtss_phy_macsec_csr_sd6g_wr (
    vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const u16 target,
    const u32 csr_reg_addr,
    u32 value )
```

Function for Writing to 6G serdes registers (applicable only on VIPER,ELISE)

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Any phy port with the chip
<i>target</i>	[IN] The CSR target
<i>csr_reg_addr</i>	[IN] The CSR register to read,
<i>value</i>	[IN] The value read

Returns

VTSS_RC_OK when new access can be done - VTSS_RC_ERROR if something went wrong and access was never granted.

8.32 vtss_api/include/vtss_phy_ts_api.h File Reference

PHY TimeStamping API.

```
#include <vtss/api/types.h>
#include <vtss/api/port.h>
#include "vtss_misc_api.h"
```

Data Structures

- struct [vtss_phy_ts_alt_clock_mode_s](#)
parameter for setting the alternative clock mode.
- struct [vtss_phy_ts_pps_config_s](#)
PPS Configuration.
- struct [vtss_phy_timestamp_t](#)
PHY timestamp in seconds and nanoseconds (10 bytes Timestamp)
- struct [vtss_phy_ts_sertod_conf_t](#)
PHY timestamp in seconds and nanoseconds (10 bytes Timestamp)
- struct [vtss_phy_ltc_freq_synth_s](#)
Frequency synthesis pulse configuration.
- struct [vtss_phy_daisy_chain_conf_t](#)
SPI daisy chain configuration.
- struct [vtss_phy_ts_fifo_sig_t](#)
Tx TSFIFO entry signature.
- struct [vtss_phy_ts_eng_init_conf_t](#)
Defines the basic engine parameters passed to the engine_init_conf_get() function.
- struct [vtss_phy_ts_eth_conf_t](#)
Analyzer Ethernet comparator configuration options.
- struct [vtss_phy_ts_ip_conf_t](#)
Analyzer IP comparator configuration options.
- struct [vtss_phy_ts_mpls_lvl_rng_t](#)
MPLS level range.
- struct [vtss_phy_ts_mpls_conf_t](#)
Analyzer MPLS comparator configuration options.
- struct [vtss_phy_ts_ach_conf_t](#)
Analyzer ACH comparator configuration options.
- struct [vtss_phy_ts_gen_conf_t](#)
Analyzer Generic data configuration options using IP comparator.
- struct [vtss_phy_ts_ptp_engine_flow_conf_t](#)
PTP engine flow configuration options.
- struct [vtss_phy_ts_oam_engine_flow_conf_t](#)
OAM engine flow configuration options.
- struct [vtss_phy_ts_generic_flow_conf_t](#)
Generic engine flow configuration options.

- struct [vtss_phy_ts_engine_flow_conf_t](#)
Analyzer flow configuration options.
- struct [vtss_phy_ts_ptp_conf_t](#)
Analyzer PTP comparator configuration options.
- struct [vtss_phy_ts_ptp_engine_action_t](#)
Analyzer PTP action configuration options.
- struct [vtss_phy_ts_y1731_oam_conf_t](#)
Analyzer OAM comparator, Y.1731 OAM Packet format configuration options.
- struct [vtss_phy_ts_ietf_mpls_ach_oam_conf_t](#)
Analyzer OAM comparator, IETF MPLS ACH OAM Packet format configuration options.
- struct [vtss_phy_ts_oam_engine_action_t](#)
OAM Action configuration options.
- struct [vtss_phy_ts_generic_action_t](#)
Generic Action configuration option.
- struct [vtss_phy_ts_engine_action_t](#)
Engine Action configuration options.
- struct [vtss_phy_ts_stats_t](#)
Timestamping Statistics.
- struct [vtss_phy_ts_init_conf_t](#)
Defines the initial parameters to be passed to init function.
- struct [vtss_phy_ts_nphase_status_t](#)
n-phase status
- struct [vtss_phy_10g_fifo_sync_t](#)
10G OOS workaround options
- struct [vtss_phy_ts_fifo_conf_t](#)
Defines the params for FIFO SYNC function.

Macros

- #define [VTSS_PHY_TS_FIFO_SIG_SRC_IP](#) 0x01
Defines Tx TSFIFO signature mask.
- #define [VTSS_PHY_TS_FIFO_SIG_DEST_IP](#) 0x02
- #define [VTSS_PHY_TS_FIFO_SIG_MSG_TYPE](#) 0x04
- #define [VTSS_PHY_TS_FIFO_SIG_DOMAIN_NUM](#) 0x08
- #define [VTSS_PHY_TS_FIFO_SIG_SOURCE_PORT_ID](#) 0x10
- #define [VTSS_PHY_TS_FIFO_SIG_SEQ_ID](#) 0x20
- #define [VTSS_PHY_TS_FIFO_SIG_DEST_MAC](#) 0x40
- #define [VTSS_PHY_TS_SIG_LEN](#) 16
- #define [VTSS_PHY_TS_SIG_TIME_STAMP_LEN](#) 10
- #define [VTSS_PHY_TS_SIG_DOMAIN_NUM_LEN](#) 1
- #define [VTSS_PHY_TS_SIG_SEQ_ID_LEN](#) 1
- #define [VTSS_PHY_TS_SIG_MSG_TYPE_LEN](#) 1
- #define [VTSS_PHY_TS_SIG_SOURCE_PORT_ID_LEN](#) 10
- #define [VTSS_PHY_TS_SIG_SEQUENCE_ID_LEN](#) 2
- #define [VTSS_PHY_TS_SIG_DEST_IP_LEN](#) 4
- #define [VTSS_PHY_TS_SIG_SRC_IP_LEN](#) 4
- #define [VTSS_PHY_TS_SIG_DEST_MAC_LEN](#) 6
- #define [VTSS_PTP_IP_1588_VERSION_2_1](#) 0x21
Defines 1588 version code for Gen-2.1.
- #define [VTSS_PHY_TS_ETH_ADDR_MATCH_48BIT](#) ((u8)0x01)
- #define [VTSS_PHY_TS_ETH_ADDR_MATCH_ANY_UNICAST](#) ((u8)0x02)

- #define VTSS_PHY_TS_ETH_ADDR_MATCH_ANY_MULTICAST ((u8)0x04)
- #define VTSS_PHY_TS_ETH_MATCH_DEST_ADDR ((u8)0x00)
- #define VTSS_PHY_TS_ETH_MATCH_SRC_ADDR ((u8)0x01)
- #define VTSS_PHY_TS_ETH_MATCH_SRC_OR_DEST ((u8)0x02)
- #define VTSS_PHY_TS_TAG_TYPE_C ((u8)0x01)
- #define VTSS_PHY_TS_TAG_TYPE_S ((u8)0x02)
- #define VTSS_PHY_TS_TAG_TYPE_I ((u8)0x03)
- #define VTSS_PHY_TS_TAG_TYPE_B ((u8)0x04)
- #define VTSS_PHY_TS_TAG_RANGE_NONE ((u8)0x00)
- #define VTSS_PHY_TS_TAG_RANGE_OUTER ((u8)0x01)
- #define VTSS_PHY_TS_TAG_RANGE_INNER ((u8)0x02)
- #define VTSS_PHY_TS_IP_VER_4 0x01
- #define VTSS_PHY_TS_IP_VER_6 0x02
- #define VTSS_PHY_TS_IP_MATCH_SRC 0x00
- #define VTSS_PHY_TS_IP_MATCH_DEST 0x01
- #define VTSS_PHY_TS_IP_MATCH_SRC_OR_DEST 0x02
- #define VTSS_PHY_TS_MPLS_STACK_DEPTH_1 ((u8)0x01)
- #define VTSS_PHY_TS_MPLS_STACK_DEPTH_2 ((u8)0x02)
- #define VTSS_PHY_TS_MPLS_STACK_DEPTH_3 ((u8)0x04)
- #define VTSS_PHY_TS_MPLS_STACK_DEPTH_4 ((u8)0x08)
- #define VTSS_PHY_TS_MPLS_STACK_REF_POINT_TOP 0x00
- #define VTSS_PHY_TS_MPLS_STACK_REF_POINT_END 0x01
- #define VTSS_PHY_TS_ENG_FLOW_VALID_FOR_CH0 0x01

Port to flow mapping within analyzer engine.

- #define VTSS_PHY_TS_ENG_FLOW_VALID_FOR_CH1 0x02
- #define VTSS_PHY_TS_INGR_ENGINE_ERR 0x01

Timestamp interrupt events.

- #define VTSS_PHY_TS_INGR_RW_PREAM_ERR 0x02
- #define VTSS_PHY_TS_INGR_RW_FCS_ERR 0x04
- #define VTSS_PHY_TS_EGR_ENGINE_ERR 0x08
- #define VTSS_PHY_TS_EGR_RW_FCS_ERR 0x10
- #define VTSS_PHY_TS_EGR_TIMESTAMP_CAPTURED 0x20
- #define VTSS_PHY_TS_EGR_FIFO_OVERFLOW 0x40
- #define VTSS_PHY_TS_DATA_IN_RSRVD_FIELD 0x80
- #define VTSS_PHY_TS_LTC_NEW_PPS_INTRPT 0x100
- #define VTSS_PHY_TS_LTC_LOAD_SAVE_NEW_TOD 0x200
- #define VTSS_PHY_TS_8487_XAUI_SEL_0 0x01

Define the Channel selection for 8487.

- #define VTSS_PHY_TS_8487_XAUI_SEL_1 0x02
- #define VTSS_PHY_TS_INGR_DATAPATH_RESET 0x01

1588 block reset

- #define VTSS_PHY_TS_EGR_DATAPATH_RESET 0x02
- #define VTSS_PHY_TS_INGR_LTC1_RESET 0x04
- #define VTSS_PHY_TS_EGR_LTC2_RESET 0x08
- #define VTSS_PHY_TS_EGR_FIFO_RESET 0x10

Typedefs

- **typedef struct vtss_phy_ts_alt_clock_mode_s vtss_phy_ts_alt_clock_mode_t**
parameter for setting the alternative clock mode.
- **typedef struct vtss_phy_ts_pps_config_s vtss_phy_ts_pps_conf_t**
PPS Configuration.
- **typedef i64 vtss_phy_ts_scaled_ppb_t**
Data type defines the clock frequency ratio in scaled ppb.
- **typedef struct vtss_phy_ltc_freq_synth_s vtss_phy_ltc_freq_synth_t**
Frequency synthesis pulse configuration.
- **typedef u32 vtss_phy_ts_fifo_sig_mask_t**
- **typedef void(* vtss_phy_ts_fifo_read) (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_timestamp_t *const ts, const vtss_phy_ts_fifo_sig_t *const sig, void *ctxt, const vtss_phy_ts_fifo_status_t status)**
Tx TSFIFO read callback function prototype.
- **typedef u8 vtss_phy_ts_engine_channel_map_t**
- **typedef u32 vtss_phy_ts_event_t**
- **typedef u32 vtss_phy_ts_8487_xaui_sel_t**
- **typedef u32 vtss_phy_ts_soft_reset_t**

Enumerations

- **enum vtss_phy_ts_todadj_status_t { VTSS_PHY_TS_TODADJ_INPROGRESS, VTSS_PHY_TS_TODADJ_DONE, VTSS_PHY_TS_TODADJ_FAIL }**
parameter describing various Tx TSFIFO status.
- **enum vtss_phy_ts_fifo_status_t { VTSS_PHY_TS_FIFO_SUCCESS, VTSS_PHY_TS_FIFO_OVERFLOW }**
parameter describing various Tx TSFIFO status.
- **enum vtss_phy_ts_encap_t { VTSS_PHY_TS_ENCAP_ETH_PTP, VTSS_PHY_TS_ENCAP_ETH_IP_PTP, VTSS_PHY_TS_ENCAP_↔ETH_IP_IP_PTP, VTSS_PHY_TS_ENCAP_ETH_ETH_PTP, VTSS_PHY_TS_ENCAP_ETH_MPLS_IP_PTP, VTSS_PHY_↔TS_ENCAP_ETH_MPLS_ETH_PTP, VTSS_PHY_TS_ENCAP_ETH_MPLS_ETH_OAM, VTSS_PHY_T↔S_ENCAP_ETH_ETH_OAM, VTSS_PHY_TS_ENCAP_ETH_MPLS_ETH_OAM, VTSS_PHY_TS_ENCAP_ETH_MPLS_ACH_OAM, VTSS_PHY_TS_ENCAP_ANY, VTSS_PHY_TS_EN↔CAP_ETH_GEN, VTSS_PHY_TS_ENCAP_NONE }**
Analyzer supported frame encapsulation type.
- **enum vtss_phy_ts_engine_t { VTSS_PHY_TS_PTP_ENGINE_ID_0, VTSS_PHY_TS_PTP_ENGINE_ID_1, VTSS_PHY_TS_OAM_ENGINE_ID_2A, VTSS_PHY_TS_OAM_ENGINE_ID_2B, VTSS_PHY_TS_ENGINE_ID_INVALID }**
Defines Analyzer engine ID.
- **enum vtss_phy_ts_engine_flow_match_t { VTSS_PHY_TS_ENG_FLOW_MATCH_ANY, VTSS_PHY_TS_ENG_FLOW_MATCH_...**
Flow matching within an analyzer engine.
- **enum vtss_phy_ts_ptp_clock_mode_t { VTSS_PHY_TS_PTP_CLOCK_MODE_BC1STEP, VTSS_PHY_TS_PTP_CLOCK_MODE_BC2STEP, VTSS_PHY_TS_PTP_CLOCK_MODE_TC1STEP, VTSS_PHY_TS_PTP_CLOCK_MODE_TC2STEP, VTSS_PHY_TS_PTP_DELAY_COMP_ENGINE }**
PTP Timestamp Engine operational modes.
- **enum vtss_phy_ts_ptp_delaym_type_t { VTSS_PHY_TS_PTP_DELAYM_P2P, VTSS_PHY_TS_PTP_DELAYM_E2E }**
PTP delay measurement method.

- enum `vtss_phy_ts_y1731_oam_delaym_type_t` { `VTSS_PHY_TS_Y1731_OAM_DELAYM_1DM`, `VTSS_PHY_TS_Y1731_OAM_DELAYM_2DM` }

Y.1731 OAM delay measurement method.

- enum `vtss_phy_ts_ietf_mpls_ach_oam_delaym_type_t` { `VTSS_PHY_TS_IETF_MPLS_ACH_OAM_DELAYM_DMM`, `VTSS_PHY_TS_IETF_MPLS_ACH_OAM_DELAYM_LDM` }

IETF MPLS ACH, OAM delay measurement method.

- enum `vtss_phy_ts_ietf_mpls_ach_oam_ts_format_t` { `VTSS_PHY_TS_IETF_MPLS_ACH_OAM_TS_FORMAT_PTP` = `0x3` }

Analyzer OAM comparator, IETF MPLS ACH OAM Supported Timestamp Format.

- enum `vtss_phy_ts_action_format` { `VTSS_PHY_TS_4_BYTE`, `VTSS_PHY_TS_10_BYTE` }

Timestamp format to be configured in action configuration.

- enum `vtss_phy_ts_clockfreq_t` { `VTSS_PHY_TS_CLOCK_FREQ_125M`, `VTSS_PHY_TS_CLOCK_FREQ_15625M`, `VTSS_PHY_TS_CLOCK_FREQ_200M`, `VTSS_PHY_TS_CLOCK_FREQ_250M`, `VTSS_PHY_TS_CLOCK_FREQ_500M`, `VTSS_PHY_TS_CLOCK_FREQ_MAX` }

Timestamp block clock frequencies.

- enum `vtss_phy_ts_clock_src_t` { `VTSS_PHY_TS_CLOCK_SRC_EXTERNAL`, `VTSS_PHY_TS_CLOCK_SRC_CLIENT_RX`, `VTSS_PHY_TS_CLOCK_SRC_CLIENT_TX`, `VTSS_PHY_TS_CLOCK_SRC_LINE_RX`, `VTSS_PHY_TS_CLOCK_SRC_LINE_TX`, `VTSS_PHY_TS_CLOCK_SRC_INTERNAL` }

Clock input source.

- enum `vtss_phy_ts_rxtimestamp_pos_t` { `VTSS_PHY_TS_RX_TIMESTAMP_POS_IN_PTP`, `VTSS_PHY_TS_RX_TIMESTAMP_POS_IN_SNTP` }

Defines Rx Timestamp position inside PTP frame.

- enum `vtss_phy_ts_rxtimestamp_len_t` { `VTSS_PHY_TS_RX_TIMESTAMP_LEN_30BIT`, `VTSS_PHY_TS_RX_TIMESTAMP_LEN_32BIT` }

Defines RX Timestamp format i.e. 30bit or 32bit Rx timestamp.

- enum `vtss_phy_ts_fifo_mode_t` { `VTSS_PHY_TS_FIFO_MODE_NORMAL`, `VTSS_PHY_TS_FIFO_MODE_SPI` }

Defines Tx TSFIFO access mode.

- enum `vtss_phy_ts_fifo_timestamp_len_t` { `VTSS_PHY_TS_FIFO_TIMESTAMP_LEN_4BYTE`, `VTSS_PHY_TS_FIFO_TIMESTAMP_LEN_10BYTE` }

Defines 4 bytes vs. the default 10 bytes Timestamp stored in Tx TSFIFO.

- enum `vtss_phy_ts_tc_op_mode_t` { `VTSS_PHY_TS_TC_OP_MODE_A` = `1`, `VTSS_PHY_TS_TC_OP_MODE_B` = `0`, `VTSS_PHY_TS_TC_OP_MODE_C` = `2` }

Defines the Transparent Clock Operating Mode.

- enum `vtss_phy_ts_nphase_sampler_t` { `VTSS_PHY_TS_NPHASE_PPS_O`, `VTSS_PHY_TS_NPHASE_PPS_RI`, `VTSS_PHY_TS_NPHASE_EGR_SOF`, `VTSS_PHY_TS_NPHASE_ING_SOF`, `VTSS_PHY_TS_NPHASE_LS`, `VTSS_PHY_TS_NPHASE_MAX` }

enum for n-phase samplers

- enum `vtss_phy_ts_ptp_message_type_t` { `PTP_SYNC_MSG`, `PTP_DELAY_REQ_MSG`, `PTP_PDELAY_REQ_MSG`, `PTP_PDELAY_RESP_MSG` }

PTP Event Message TYPES.

Functions

- `vtss_rc vtss_phy_ts_alt_clock_saved_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `u32` *const saved)

Get the latest saved nanosec counter from the alternative clock.

- `vtss_rc vtss_phy_ts_alt_clock_mode_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_ts_alt_clock_mode_t` *const phy_alt_clock_mode)

- `vtss_rc vtss_phy_ts_alt_clock_mode_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_phy_ts_alt_clock_mode_t` *const phy_alt_clock_mode)

Set the alternative clock mode. This function configures the loopbacks.
- `vtss_rc vtss_phy_ts_pps_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_phy_ts_pps_conf_t` *const phy_pps_conf)

Set offset for the PPS generation.
- `vtss_rc vtss_phy_ts_pps_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_ts_pps_conf_t` *const phy_pps_conf)

Get offset for the PPS generation.
- `vtss_rc vtss_phy_ts_ingress_latency_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_timeinterval_t` *const latency)

Set the ingress latency.
- `vtss_rc vtss_phy_ts_ingress_latency_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_timeinterval_t` *const latency)

Get the ingress latency.
- `vtss_rc vtss_phy_ts_egress_latency_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_timeinterval_t` *const latency)

Set the egress latency.
- `vtss_rc vtss_phy_ts_egress_latency_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_timeinterval_t` *const latency)

Get the egress latency.
- `vtss_rc vtss_phy_ts_path_delay_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_timeinterval_t` *const path_delay)

Set the path delay.
- `vtss_rc vtss_phy_ts_path_delay_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_timeinterval_t` *const path_delay)

Get the path delay.
- `vtss_rc vtss_phy_ts_delay_asymmetry_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_timeinterval_t` *const delay_asym)

Set the delay asymmetry.
- `vtss_rc vtss_phy_ts_delay_asymmetry_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_timeinterval_t` *const delay_asym)

Get the delay asymmetry.
- `vtss_rc vtss_phy_ts_ptptime_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_phy_timestamp_t` *const ts)

Set the current PTP time into the PHY.
- `vtss_rc vtss_phy_ts_ptptime_set_done` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no)

Setting of the current PTP time into the PHY is completed.
- `vtss_rc vtss_phy_ts_ptptime_arm` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no)

Arm the local time of the PHY so that in next pps it can be read.
- `vtss_rc vtss_phy_ts_ptptime_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_timestamp_t` *const ts)

Get the armed PTP time from the PHY.
- `vtss_rc vtss_phy_ts_load_ptptime_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_timestamp_t` *const ts)

Get the PTP time from the PHY load registers.
- `vtss_rc vtss_phy_ts_sertod_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_phy_ts_sertod_conf_t` *const sertod_conf)

Set Enable/Disable Serial ToD.
- `vtss_rc vtss_phy_ts_sertod_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_ts_sertod_conf_t` *const sertod_conf)

Get Enable/Disable Serial ToD.

- `vtss_rc vtss_phy_ts_loadpulse_delay_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `u16` *const delay)

Set load pulse delay.
- `vtss_rc vtss_phy_ts_loadpulse_delay_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `u16` *const delay)

Get load pulse delay.
- `vtss_rc vtss_phy_ts_clock_rateadj_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_phy_ts_scaled_ppb_t` *const adj)

Adjust the local clock rate.
- `vtss_rc vtss_phy_ts_clock_rateadj_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_ts_scaled_ppb_t` *const adj)

Get the clock rate adjustment value.
- `vtss_rc vtss_phy_ts_clock_rateadj_ppm_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_phy_ts_scaled_ppb_t` *const adj)

Adjust ppm of the local clock rate .
- `vtss_rc vtss_phy_ts_clock_rateadj_ppm_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_ts_scaled_ppb_t` *const adj)

Get the clock rate ppm adjustment value.
- `vtss_rc vtss_phy_ts_ptptime_adj1ns` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `BOOL` incr)

Increment/decrement the LTC clock value by 1 ns.
- `vtss_rc vtss_phy_ts_timeofday_offset_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `i32` offset)

Subtract offset from the current time.
- `vtss_rc vtss_phy_ts_ongoing_adjustment` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_ts_todadj_status_t` *const ongoing_adjustment)

Return the status of the LTC time adjustment.
- `vtss_rc vtss_phy_ts_ltc_freq_synth_pulse_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_phy_ltc_freq_synth_t` *const ltc_freq_synthesis)

Return the status of the LTC time adjustment.
- `vtss_rc vtss_phy_ts_ltc_freq_synth_pulse_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_ltc_freq_synth_t` *const ltc_freq_synthesis)

Return the status of the LTC time adjustment.
- `vtss_rc vtss_phy_daisy_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_phy_daisy_chain_conf_t` *const daisy_chain)

configure the daisy chain for TS FIFO
- `vtss_rc vtss_phy_daisy_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_daisy_chain_conf_t` *const daisy_chain)

getting the daisy chain for TS FIFO
- `vtss_rc vtss_phy_ts_fifo_sig_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_phy_ts_fifo_sig_mask_t` sig_mask)

Set Tx TSFIFO i.e. frame signature in Tx TSFIFO.
- `vtss_rc vtss_phy_ts_fifo_sig_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_ts_fifo_sig_mask_t` *const sig_mask)

Get frame signature mask in Tx TSFIFO.
- `vtss_rc vtss_phy_ts_fifo_empty` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no)

Read timestamp from Tx TSFIFO.
- `vtss_rc vtss_phy_ts_fifo_read_install` (const `vtss_inst_t` inst, `vtss_phy_ts_fifo_read` rd_cb, void *cctxt)

Install callback to read data (signature + timestamp) from Tx TSFIFO.
- `vtss_rc vtss_phy_ts_fifo_read_cb_get` (const `vtss_inst_t` inst, `vtss_phy_ts_fifo_read` *rd_cb, void **cctxt)

Get the fifo read callback function installed.
- `vtss_rc vtss_phy_ts_ingress_engine_init` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_phy_ts_engine_t` eng_id, const `vtss_phy_ts_encap_t` encaps_type, const `u8` flow_st_index, const `u8` flow_end_index, const `vtss_phy_ts_engine_flow_match_t` flow_match_mode)

Initialize an analyzer ingress engine for an encapsulation type.

- `vtss_rc vtss_phy_ts_ingress_engine_init_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_phy_ts_engine_t` eng_id, `vtss_phy_ts_eng_init_conf_t` *const init_conf)

Get the configuration parameters passed in engine_init of an analyzer ingress engine for a specific engine ID.

- `vtss_rc vtss_phy_ts_ingress_engine_clear` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_phy_ts_engine_t` eng_id)

Clear/release an analyzer ingress engine already initialized.

- `vtss_rc vtss_phy_ts_egress_engine_init` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_phy_ts_engine_t` eng_id, const `vtss_phy_ts_encap_t` encapsulation_type, const `u8` flow_st_index, const `u8` flow_end_index, const `vtss_phy_ts_engine_flow_match_t` flow_match_mode)

Initialize an analyzer egress engine for an encapsulation type.

- `vtss_rc vtss_phy_ts_egress_engine_init_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_phy_ts_engine_t` eng_id, `vtss_phy_ts_eng_init_conf_t` *const init_conf)

Get the configuration parameters passed in engine_init of an analyzer egress engine for a specific engine ID.

- `vtss_rc vtss_phy_ts_egress_engine_clear` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_phy_ts_engine_t` eng_id)

Clear/release an analyzer egress engine already initialized.

- `vtss_rc vtss_phy_ts_ingress_engine_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_phy_ts_engine_t` eng_id, const `vtss_phy_ts_engine_flow_conf_t` *const flow_conf)

Configure ingress analyzer flow.

- `vtss_rc vtss_phy_ts_ingress_engine_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_phy_ts_engine_t` eng_id, `vtss_phy_ts_engine_flow_conf_t` *const flow_conf)

Get ingress analyzer flow.

- `vtss_rc vtss_phy_ts_egress_engine_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_phy_ts_engine_t` eng_id, const `vtss_phy_ts_engine_flow_conf_t` *const flow_conf)

Configure egress analyzer flow.

- `vtss_rc vtss_phy_ts_egress_engine_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_phy_ts_engine_t` eng_id, `vtss_phy_ts_engine_flow_conf_t` *const flow_conf)

Get egress analyzer flow.

- `vtss_rc vtss_phy_ts_ingress_engine_action_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_phy_ts_engine_t` eng_id, const `vtss_phy_ts_engine_action_t` *const action_conf)

Configure ingress analyzer engine action.

- `vtss_rc vtss_phy_ts_ingress_engine_action_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_phy_ts_engine_t` eng_id, `vtss_phy_ts_engine_action_t` *const action_conf)

Get ingress analyzer engine action. For multi-port timestamp block, either of the port can be used to get the ingress engine.

- `vtss_rc vtss_phy_ts_egress_engine_action_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_phy_ts_engine_t` eng_id, const `vtss_phy_ts_engine_action_t` *const action_conf)

Configure egress analyzer engine action.

- `vtss_rc vtss_phy_ts_egress_engine_action_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_phy_ts_engine_t` eng_id, `vtss_phy_ts_engine_action_t` *const action_conf)

Get egress analyzer engine action. For multi-port timestamp block, either of the port can be used to get the egress engine.

- `vtss_rc vtss_phy_ts_event_enable_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `BOOL` enable, const `vtss_phy_ts_event_t` ev_mask)

Enabling / Disabling of events.

- `vtss_rc vtss_phy_ts_event_enable_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_ts_event_t` *const ev_mask)

Get Enabling of events.

- `vtss_rc vtss_phy_ts_event_poll` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_ts_event_t` *const status)

Polling function called at by interrupt or periodically.

- `vtss_rc vtss_phy_ts_stats_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_ts_stats_t` *const statistics)

- Get Timestamp statistics.*
- `vtss_rc vtss_phy_ts_correction_overflow_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `BOOL` *const ingr_overflow, `BOOL` *const egr_overflow)

Get the correction field overflow status in ingress and egress.

 - `vtss_rc vtss_phy_ts_mode_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `BOOL` enable)

Enable/disable timestamp block.

 - `vtss_rc vtss_phy_ts_mode_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `BOOL` *const enable)

Get timestamp block status i.e. enable/disable.

 - `vtss_rc vtss_phy_ts_init` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_phy_ts_init_conf_t` *const conf)

Init timestamp block.

 - `vtss_rc vtss_phy_ts_init_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `BOOL` *const init_done, `vtss_phy_ts_init_conf_t` *const conf)

Get the timestamp init config parameters.

 - `vtss_rc vtss_phy_ts_nphase_status_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_ts_nphase_sampler_t` sampler, `vtss_phy_ts_nphase_status_t` *const status)

Get N-Phase sampler status.

 - `vtss_rc vtss_phy_ts_hiacc_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_ts_nphase_sampler_t` sampler, const `BOOL` enable)

Enable N-Phase sampler.

 - `vtss_rc vtss_phy_ts_hiacc_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_phy_ts_nphase_sampler_t` sampler, `BOOL` const *enable)

N-Phase sampler status get.

 - `vtss_rc vtss_phy_ts_block_soft_reset` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_phy_ts_soft_reset_t` ts_reset)

reset 1588 block.

 - `vtss_rc vtss_phy_ts_new_spi_mode_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `BOOL` enable)

Enable/disable New SPI mode for 8574-15 (Rev A & Rev B) that uses PPS0 pin as the new SPI_CLK.

 - `vtss_rc vtss_phy_ts_new_spi_mode_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `BOOL` *const mode)

Get New SPI mode for 8574-15 (Rev A & Rev B) described above.

 - `vtss_rc vtss_phy_ts_phy_oper_mode_change` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no)

Update the PHY timestamping block to predict the correct latency.

 - `vtss_rc vtss_phy_1588_csr_reg_write` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `u32` blk_id, const `u16` csr_address, const `u32` *const value)

Set the the 1588 block CSR registers.

 - `vtss_rc vtss_phy_1588_csr_reg_read` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `u32` blk_id, const `u16` csr_address, `u32` *const value)

get the the 1588 block CSR registers.

 - `vtss_rc vtss_phy_ts_status_check` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `BOOL` wait, const `vtss_debug_printf_t` pr)

TS status check function supported for 10G Phys like 8488 & 8492.

 - `vtss_rc vtss_phy_ts_10g_extended_fifo_sync` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_phy_10g_fifo_sync_t` *conf)

Reset 1588 PHY Fifo oos. As of now, supported for 10G Phys like 8488.

 - `vtss_rc vtss_phy_ts_10g_fifo_sync` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_debug_printf_t` pr)

Reset 1588 PHY Fifo oos. As of now, supported for 10G Phys like 8488.

 - `vtss_rc vtss_phy_ts_bypass_clear` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_debug_printf_t` pr)

1588 Bypass clear

- `vtss_rc vtss_phy_ts_viper_fifo_reset (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_ts_fifo_conf_t *fifo_conf)`
Viper 1588 FIFO reset.
- `vtss_rc vtss_phy_ts_tesla_tsp_fifo_sync (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_debug_printf_t pr, const vtss_phy_ts_fifo_conf_t *fifo_conf, BOOL *OOS)`
API to detect and correct Timestamp FIFO OOS.
- `vtss_rc vtss_phy_1g_ts_fifo_sync (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_debug_printf_t pr, const vtss_phy_ts_fifo_conf_t *fifo_conf, BOOL *OOS)`
API to detect and correct Timestamp FIFO OOS for 1G PHY's (Viper and Tesla)
- `vtss_rc vtss_phy_1588_debug_reg_read (const vtss_inst_t inst, const vtss_port_no_t port_no, const u32 blk_id, const vtss_debug_printf_t p_routine)`
API to dump PHY timestamp registers (for Debugging)
- `vtss_rc vtss_phy_ts_flow_clear_cf_set (const vtss_inst_t inst, const vtss_port_no_t port_no, BOOL ingress, const vtss_phy_ts_engine_t eng_id, u8 act_id, vtss_phy_ts_ptp_message_type_t msgtype)`
Clear Correction field for specified PTP message type.

8.32.1 Detailed Description

PHY TimeStamping API.

This header file describes PTP/OAM TimeStamping API functions

8.32.2 Macro Definition Documentation

8.32.2.1 VTSS_PHY_TS_FIFO_SIG_SRC_IP

```
#define VTSS_PHY_TS_FIFO_SIG_SRC_IP 0x01
```

Defines Tx TSFIFO signature mask.

Src IP address: inner IP for IP-over-IP

Definition at line 570 of file vtss_phy_ts_api.h.

8.32.2.2 VTSS_PHY_TS_FIFO_SIG_DEST_IP

```
#define VTSS_PHY_TS_FIFO_SIG_DEST_IP 0x02
```

Dest IP address

Definition at line 571 of file vtss_phy_ts_api.h.

8.32.2.3 VTSS_PHY_TS_FIFO_SIG_MSG_TYPE

```
#define VTSS_PHY_TS_FIFO_SIG_MSG_TYPE 0x04
```

Message type

Definition at line 573 of file vtss_phy_ts_api.h.

8.32.2.4 VTSS_PHY_TS_FIFO_SIG_DOMAIN_NUM

```
#define VTSS_PHY_TS_FIFO_SIG_DOMAIN_NUM 0x08
```

Domain number

Definition at line 574 of file vtss_phy_ts_api.h.

8.32.2.5 VTSS_PHY_TS_FIFO_SIG_SOURCE_PORT_ID

```
#define VTSS_PHY_TS_FIFO_SIG_SOURCE_PORT_ID 0x10
```

Source port identity

Definition at line 575 of file vtss_phy_ts_api.h.

8.32.2.6 VTSS_PHY_TS_FIFO_SIG_SEQ_ID

```
#define VTSS_PHY_TS_FIFO_SIG_SEQ_ID 0x20
```

PTP frame Sequence ID

Definition at line 576 of file vtss_phy_ts_api.h.

8.32.2.7 VTSS_PHY_TS_FIFO_SIG_DEST_MAC

```
#define VTSS_PHY_TS_FIFO_SIG_DEST_MAC 0x40
```

Dest MAC address

Definition at line 578 of file vtss_phy_ts_api.h.

8.32.2.8 VTSS_PHY_TS_SIG_LEN

```
#define VTSS_PHY_TS_SIG_LEN 16
```

TS Signature length

Definition at line 586 of file vtss_phy_ts_api.h.

8.32.2.9 VTSS_PHY_TS_SIG_TIME_STAMP_LEN

```
#define VTSS_PHY_TS_SIG_TIME_STAMP_LEN 10
```

Timestamp Bytes in TS Signature

Definition at line 587 of file vtss_phy_ts_api.h.

8.32.2.10 VTSS_PHY_TS_SIG_DOMAIN_NUM_LEN

```
#define VTSS_PHY_TS_SIG_DOMAIN_NUM_LEN 1
```

Domain number length in TS Signature

Definition at line 589 of file vtss_phy_ts_api.h.

8.32.2.11 VTSS_PHY_TS_SIG_SEQ_ID_LEN

```
#define VTSS_PHY_TS_SIG_SEQ_ID_LEN 1
```

Seq ID length in TS Signature

Definition at line 590 of file vtss_phy_ts_api.h.

8.32.2.12 VTSS_PHY_TS_SIG_MSG_TYPE_LEN

```
#define VTSS_PHY_TS_SIG_MSG_TYPE_LEN 1
```

MSG Type length in TS Signature

Definition at line 591 of file vtss_phy_ts_api.h.

8.32.2.13 VTSS_PHY_TS_SIG_SOURCE_PORT_ID_LEN

```
#define VTSS_PHY_TS_SIG_SOURCE_PORT_ID_LEN 10
```

Source Port length in TS Signature

Definition at line 592 of file vtss_phy_ts_api.h.

8.32.2.14 VTSS_PHY_TS_SIG_SEQUENCE_ID_LEN

```
#define VTSS_PHY_TS_SIG_SEQUENCE_ID_LEN 2
```

Sequence ID length in TS Signature

Definition at line 593 of file vtss_phy_ts_api.h.

8.32.2.15 VTSS_PHY_TS_SIG_DEST_IP_LEN

```
#define VTSS_PHY_TS_SIG_DEST_IP_LEN 4
```

Dest IP length in TS Signature

Definition at line 594 of file vtss_phy_ts_api.h.

8.32.2.16 VTSS_PHY_TS_SIG_SRC_IP_LEN

```
#define VTSS_PHY_TS_SIG_SRC_IP_LEN 4
```

SRC IP length in TS Signature

Definition at line 595 of file vtss_phy_ts_api.h.

8.32.2.17 VTSS_PHY_TS_SIG_DEST_MAC_LEN

```
#define VTSS_PHY_TS_SIG_DEST_MAC_LEN 6
```

Dest MAC length in TS Signature

Definition at line 596 of file vtss_phy_ts_api.h.

8.32.2.18 VTSS_PTP_IP_1588_VERSION_2_1

```
#define VTSS_PTP_IP_1588_VERSION_2_1 0x21
```

Defines 1588 version code for Gen-2.1.

1588 block version for Gen2.1

Definition at line 605 of file vtss_phy_ts_api.h.

8.32.2.19 VTSS_PHY_TS_ETH_ADDR_MATCH_48BIT

```
#define VTSS_PHY_TS_ETH_ADDR_MATCH_48BIT ((u8)0x01)
```

Full 48-bit address match

Definition at line 965 of file vtss_phy_ts_api.h.

8.32.2.20 VTSS_PHY_TS_ETH_ADDR_MATCH_ANY_UNICAST

```
#define VTSS_PHY_TS_ETH_ADDR_MATCH_ANY_UNICAST ((u8)0x02)
```

Match any unicast MAC address

Definition at line 966 of file vtss_phy_ts_api.h.

8.32.2.21 VTSS_PHY_TS_ETH_ADDR_MATCH_ANY_MULTICAST

```
#define VTSS_PHY_TS_ETH_ADDR_MATCH_ANY_MULTICAST ((u8)0x04)
```

Match any multicast MAC address

Definition at line 967 of file vtss_phy_ts_api.h.

8.32.2.22 VTSS_PHY_TS_ETH_MATCH_DEST_ADDR

```
#define VTSS_PHY_TS_ETH_MATCH_DEST_ADDR ((u8)0x00)
```

Match destination MAC address

Definition at line 969 of file vtss_phy_ts_api.h.

8.32.2.23 VTSS_PHY_TS_ETH_MATCH_SRC_ADDR

```
#define VTSS_PHY_TS_ETH_MATCH_SRC_ADDR ((u8)0x01)
```

Match source MAC address

Definition at line 970 of file vtss_phy_ts_api.h.

8.32.2.24 VTSS_PHY_TS_ETH_MATCH_SRC_OR_DEST

```
#define VTSS_PHY_TS_ETH_MATCH_SRC_OR_DEST ((u8)0x02)
```

Match source or destination MAC address

Definition at line 971 of file vtss_phy_ts_api.h.

8.32.2.25 VTSS_PHY_TS_TAG_TYPE_C

```
#define VTSS_PHY_TS_TAG_TYPE_C ((u8)0x01)
```

Tag type: C

Definition at line 977 of file vtss_phy_ts_api.h.

8.32.2.26 VTSS_PHY_TS_TAG_TYPE_S

```
#define VTSS_PHY_TS_TAG_TYPE_S ((u8)0x02)
```

Tag type: S

Definition at line 978 of file vtss_phy_ts_api.h.

8.32.2.27 VTSS_PHY_TS_TAG_TYPE_I

```
#define VTSS_PHY_TS_TAG_TYPE_I ((u8)0x03)
```

Tag type: I

Definition at line 979 of file vtss_phy_ts_api.h.

8.32.2.28 VTSS_PHY_TS_TAG_TYPE_B

```
#define VTSS_PHY_TS_TAG_TYPE_B ((u8) 0x04)
```

Tag type: B

Definition at line 980 of file vtss_phy_ts_api.h.

8.32.2.29 VTSS_PHY_TS_TAG_RANGE_NONE

```
#define VTSS_PHY_TS_TAG_RANGE_NONE ((u8) 0x00)
```

Neither inner nor outer tag allows range config

Definition at line 983 of file vtss_phy_ts_api.h.

8.32.2.30 VTSS_PHY_TS_TAG_RANGE_OUTER

```
#define VTSS_PHY_TS_TAG_RANGE_OUTER ((u8) 0x01)
```

Outer tag allows range config

Definition at line 984 of file vtss_phy_ts_api.h.

8.32.2.31 VTSS_PHY_TS_TAG_RANGE_INNER

```
#define VTSS_PHY_TS_TAG_RANGE_INNER ((u8) 0x02)
```

Inner tag allows range config

Definition at line 985 of file vtss_phy_ts_api.h.

8.32.2.32 VTSS_PHY_TS_IP_VER_4

```
#define VTSS_PHY_TS_IP_VER_4 0x01
```

Version: IPv4

Definition at line 1021 of file vtss_phy_ts_api.h.

8.32.2.33 VTSS_PHY_TS_IP_VER_6

```
#define VTSS_PHY_TS_IP_VER_6 0x02
```

Version: IPv6

Definition at line 1022 of file vtss_phy_ts_api.h.

8.32.2.34 VTSS_PHY_TS_IP_MATCH_SRC

```
#define VTSS_PHY_TS_IP_MATCH_SRC 0x00
```

Match source IP address

Definition at line 1033 of file vtss_phy_ts_api.h.

8.32.2.35 VTSS_PHY_TS_IP_MATCH_DEST

```
#define VTSS_PHY_TS_IP_MATCH_DEST 0x01
```

Match destination IP address

Definition at line 1034 of file vtss_phy_ts_api.h.

8.32.2.36 VTSS_PHY_TS_IP_MATCH_SRC_OR_DEST

```
#define VTSS_PHY_TS_IP_MATCH_SRC_OR_DEST 0x02
```

Match source or destination IP address

Definition at line 1035 of file vtss_phy_ts_api.h.

8.32.2.37 VTSS_PHY_TS_MPLS_STACK_DEPTH_1

```
#define VTSS_PHY_TS_MPLS_STACK_DEPTH_1 ((u8)0x01)
```

MPLS stack of depth 1 only allows

Definition at line 1068 of file vtss_phy_ts_api.h.

8.32.2.38 VTSS_PHY_TS_MPLS_STACK_DEPTH_2

```
#define VTSS_PHY_TS_MPLS_STACK_DEPTH_2 ((u8)0x02)
```

MPLS stack of depth 2 only allows

Definition at line 1069 of file vtss_phy_ts_api.h.

8.32.2.39 VTSS_PHY_TS_MPLS_STACK_DEPTH_3

```
#define VTSS_PHY_TS_MPLS_STACK_DEPTH_3 ((u8)0x04)
```

MPLS stack of depth 3 only allows

Definition at line 1070 of file vtss_phy_ts_api.h.

8.32.2.40 VTSS_PHY_TS_MPLS_STACK_DEPTH_4

```
#define VTSS_PHY_TS_MPLS_STACK_DEPTH_4 ((u8)0x08)
```

MPLS stack of depth 4 only allows

Definition at line 1071 of file vtss_phy_ts_api.h.

8.32.2.41 VTSS_PHY_TS_MPLS_STACK_REF_POINT_TOP

```
#define VTSS_PHY_TS_MPLS_STACK_REF_POINT_TOP 0x00
```

Search starts from the top of the stack

Definition at line 1074 of file vtss_phy_ts_api.h.

8.32.2.42 VTSS_PHY_TS_MPLS_STACK_REF_POINT_END

```
#define VTSS_PHY_TS_MPLS_STACK_REF_POINT_END 0x01
```

Search starts from the end of the stack

Definition at line 1075 of file vtss_phy_ts_api.h.

8.32.2.43 VTSS_PHY_TS_ENG_FLOW_VALID_FOR_CH0

```
#define VTSS_PHY_TS_ENG_FLOW_VALID_FOR_CH0 0x01
```

Port to flow mapping within analyzer engine.

Note

This is applicable for multi-channel timestamp block.Channel-0 mapped

Definition at line 1139 of file vtss_phy_ts_api.h.

8.32.2.44 VTSS_PHY_TS_ENG_FLOW_VALID_FOR_CH1

```
#define VTSS_PHY_TS_ENG_FLOW_VALID_FOR_CH1 0x02
```

Channel-1 mapped

Definition at line 1140 of file vtss_phy_ts_api.h.

8.32.2.45 VTSS_PHY_TS_INGR_ENGINE_ERR

```
#define VTSS_PHY_TS_INGR_ENGINE_ERR 0x01
```

Timestamp interrupt events.

More than one engine find match

Definition at line 1508 of file vtss_phy_ts_api.h.

8.32.2.46 VTSS_PHY_TS_INGR_RW_PREAM_ERR

```
#define VTSS_PHY_TS_INGR_RW_PREAM_ERR 0x02
```

Preamble too short to append timestamp

Definition at line 1509 of file vtss_phy_ts_api.h.

8.32.2.47 VTSS_PHY_TS_INGR_RW_FCS_ERR

```
#define VTSS_PHY_TS_INGR_RW_FCS_ERR 0x04
```

FCS error in ingress

Definition at line 1510 of file vtss_phy_ts_api.h.

8.32.2.48 VTSS_PHY_TS_EGR_ENGINE_ERR

```
#define VTSS_PHY_TS_EGR_ENGINE_ERR 0x08
```

More than one engine find match

Definition at line 1511 of file vtss_phy_ts_api.h.

8.32.2.49 VTSS_PHY_TS_EGR_RW_FCS_ERR

```
#define VTSS_PHY_TS_EGR_RW_FCS_ERR 0x10
```

FCS error in egress

Definition at line 1512 of file vtss_phy_ts_api.h.

8.32.2.50 VTSS_PHY_TS_EGR_TIMESTAMP_CAPTURED

```
#define VTSS_PHY_TS_EGR_TIMESTAMP_CAPTURED 0x20
```

Timestamp captured in Tx TSFIFO

Definition at line 1513 of file vtss_phy_ts_api.h.

8.32.2.51 VTSS_PHY_TS_EGR_FIFO_OVERFLOW

```
#define VTSS_PHY_TS_EGR_FIFO_OVERFLOW 0x40
```

Tx TSFIFO overflow

Definition at line 1514 of file vtss_phy_ts_api.h.

8.32.2.52 VTSS_PHY_TS_DATA_IN_RSRVD_FIELD

```
#define VTSS_PHY_TS_DATA_IN_RSRVD_FIELD 0x80
```

Data in reserved Field

Definition at line 1515 of file vtss_phy_ts_api.h.

8.32.2.53 VTSS_PHY_TS_LTC_NEW_PPS_INTRPT

```
#define VTSS_PHY_TS_LTC_NEW_PPS_INTRPT 0x100
```

New PPS pushed onto external PPS pin

Definition at line 1516 of file vtss_phy_ts_api.h.

8.32.2.54 VTSS_PHY_TS_LTC_LOAD_SAVE_NEW_TOD

```
#define VTSS_PHY_TS_LTC_LOAD_SAVE_NEW_TOD 0x200
```

New LTC value either loaded in to HW or saved into registers

Definition at line 1517 of file vtss_phy_ts_api.h.

8.32.2.55 VTSS_PHY_TS_8487_XAUI_SEL_0

```
#define VTSS_PHY_TS_8487_XAUI_SEL_0 0x01
```

Define the Channel selection for 8487.

Select XAUI Lane - 0

Definition at line 1736 of file vtss_phy_ts_api.h.

8.32.2.56 VTSS_PHY_TS_8487_XAUI_SEL_1

```
#define VTSS_PHY_TS_8487_XAUI_SEL_1 0x02
```

Select XAUI Lane - 1

Definition at line 1737 of file vtss_phy_ts_api.h.

8.32.2.57 VTSS_PHY_TS_INGR_DATAPATH_RESET

```
#define VTSS_PHY_TS_INGR_DATAPATH_RESET 0x01
```

1588 block reset

chip's ingress data path in the 1588 processing block

Definition at line 1938 of file vtss_phy_ts_api.h.

8.32.2.58 VTSS_PHY_TS_EGR_DATAPATH_RESET

```
#define VTSS_PHY_TS_EGR_DATAPATH_RESET 0x02
```

chip's egress data path in the 1588 processing block

Definition at line 1939 of file vtss_phy_ts_api.h.

8.32.2.59 VTSS_PHY_TS_INGR_LTC1_RESET

```
#define VTSS_PHY_TS_INGR_LTC1_RESET 0x04
```

Ingress LTC clock domain logic for this channel

Definition at line 1940 of file vtss_phy_ts_api.h.

8.32.2.60 VTSS_PHY_TS_EGR_LTC2_RESET

```
#define VTSS_PHY_TS_EGR_LTC2_RESET 0x08
```

Egress LTC clock domain logic for this channel

Definition at line 1941 of file vtss_phy_ts_api.h.

8.32.2.61 VTSS_PHY_TS_EGR_FIFO_RESET

```
#define VTSS_PHY_TS_EGR_FIFO_RESET 0x10
```

Egress FIFO reset

Definition at line 1942 of file vtss_phy_ts_api.h.

8.32.3 Typedef Documentation

8.32.3.1 `vtss_phy_ts_alt_clock_mode_t`

```
typedef struct vtss_phy_ts_alt_clock_mode_s vtss_phy_ts_alt_clock_mode_t
```

parameter for setting the alternative clock mode.

external clock output configuration.

8.32.3.2 `vtss_phy_ts_scaled_ppb_t`

```
typedef i64 vtss_phy_ts_scaled_ppb_t
```

Data type defines the clock frequency ratio in scaled ppb.

Note

The frequency of the internal clock can be adjusted in units of scaledPartsPerBillion, which is defined as the rate in units of ppb and multiplied by 2^{16} and contained in a signed 64 bit value. For example, 2.5 ppb is expressed as 0000 0000 0002 8000

Definition at line 393 of file vtss_phy_ts_api.h.

8.32.3.3 `vtss_phy_ts_fifo_sig_mask_t`

```
typedef u32 vtss_phy_ts_fifo_sig_mask_t
```

Signature mask which can be OR of multiple fields above

Definition at line 584 of file vtss_phy_ts_api.h.

8.32.3.4 `vtss_phy_ts_fifo_read`

```
typedef void(* vtss_phy_ts_fifo_read) (const vtss_inst_t inst, const vtss_port_no_t port_no,
                                         const vtss_phy_timestamp_t *const ts, const vtss_phy_ts_fifo_sig_t *const sig, void *cntxt,
                                         const vtss_phy_ts_fifo_status_t status)
```

Tx TSFIFO read callback function prototype.

Tx TSFIFO API to access the HW TXFIFO. Application has to install the callback function which is called to push timestamp from the HW TXFIFO to the application. inst handle to an API instance port_no port number ts captured timestamp sig timestamp signature cntxt context to be returned in callback status FIFO read status

Definition at line 696 of file vtss_phy_ts_api.h.

8.32.3.5 vtss_phy_ts_engine_channel_map_t

```
typedef u8 vtss_phy_ts_engine_channel_map_t
```

Channel-0 or channel-1 or both the channels

Definition at line 1141 of file vtss_phy_ts_api.h.

8.32.3.6 vtss_phy_ts_event_t

```
typedef u32 vtss_phy_ts_event_t
```

Int events: Single event or 'OR' multiple events above

Definition at line 1519 of file vtss_phy_ts_api.h.

8.32.3.7 vtss_phy_ts_8487_xaui_sel_t

```
typedef u32 vtss_phy_ts_8487_xaui_sel_t
```

XAUI Lane-0 or Lane-1 or both

Definition at line 1738 of file vtss_phy_ts_api.h.

8.32.3.8 vtss_phy_ts_soft_reset_t

```
typedef u32 vtss_phy_ts_soft_reset_t
```

Reset blocks: Single or 'OR' multiple above

Definition at line 1944 of file vtss_phy_ts_api.h.

8.32.4 Enumeration Type Documentation

8.32.4.1 vtss_phy_ts_todadj_status_t

```
enum vtss_phy_ts_todadj_status_t
```

parameter describing various Tx TSFIFO status.

Enumerator

VTSS_PHY_TS_TODADJ_INPROGRESS	ToD Adjustment is in progress
VTSS_PHY_TS_TODADJ_DONE	ToD Adjustment is completed
VTSS_PHY_TS_TODADJ_FAIL	ToD Adjustment Failed

Definition at line 477 of file vtss_phy_ts_api.h.

8.32.4.2 vtss_phy_ts_fifo_status_t

enum [vtss_phy_ts_fifo_status_t](#)

parameter describing various Tx TSFIFO status.

Following Tx TSFIFO related API are used if FIFO access mode is set as PHY_TS_FIFO_MODE_NORMAL. In SPI mode, timestamps are pushed into SPI interface as soon as they are available.

Enumerator

VTSS_PHY_TS_FIFO_SUCCESS	FIFO read success
VTSS_PHY_TS_FIFO_OVERFLOW	FIFO overflow

Definition at line 648 of file vtss_phy_ts_api.h.

8.32.4.3 vtss_phy_ts_encap_t

enum [vtss_phy_ts_encap_t](#)

Analyzer supported frame encapsulation type.

Analyzer API

Definition at line 738 of file vtss_phy_ts_api.h.

8.32.4.4 vtss_phy_ts_engine_t

enum [vtss_phy_ts_engine_t](#)

Defines Analyzer engine ID.

Note

Timestamp block has 2 PTP engines and 1 OAM engine. OAM engine has two sub-engines (each supports different frame encapsulation) which share OAM comparator to config time stamp functionality. API will expose these 2 sub-engines to the application as 2 independent engines which can have common/shared time stamping functionality (we call it as action). So API will provide 2 PTP and 2 OAM engines to the application to use with following properties/restriction which application must remember while programming an engine.

- (1) Multi-port timestamp block can share the same engine for both the ports where for each flow in the engine application has to mention flow belong to either one of the ports or both the ports; we call it as channel map.
- (2) Each PTP engine supports 8 flows in ETH, IP and MPLS comparators and 6 actions in PTP/OAM comparator. OAM engines (2A and 2B) have total of 8 flows and 6 actions. Application has to associate flows to OAM engines. But OAM actions can be shared between the 2 OAM engines.
- (3) There is one HW limitation for flow match mode (strict/non-strict). For same engine ID in ingress and egress direction flow match mode must be same i.e. if engine VTSS_PHY_TS_PTP_ENGINE_ID_0 in ingress is configured as strict flow match then engine VTSS_PHY_TS_PTP_ENGINE_ID_0 in egress has to be in strict flow match. Also OAM engine 2A and 2B can not have different flow match mode i.e engine 2A and 2B for ingress and egress must have same flow match mode.
- (4) OAM engine does not support TSFIFO and it can not be used for PTP application. But OAM application can use PTP engine.
- (5) OAM engine 2B only support OAM application with single ethernet encap i.e. OAM-over-ETH

Enumerator

VTSS_PHY_TS_PTP_ENGINE_ID_0	PTP engine 0
VTSS_PHY_TS_PTP_ENGINE_ID_1	PTP engine 1
VTSS_PHY_TS_OAM_ENGINE_ID_2A	OAM engine 2A, no PTP support
VTSS_PHY_TS_OAM_ENGINE_ID_2B	OAM engine 2B, no PTP; only OAM-over-ETH support
VTSS_PHY_TS_ENGINE_ID_INVALID	Invalid Engine ID

Definition at line 791 of file vtss_phy_ts_api.h.

8.32.4.5 vtss_phy_ts_engine_flow_match_t

```
enum vtss_phy_ts_engine_flow_match_t
```

Flow matching within an analyzer engine.

Note

There are two types of flow match possible:

- (1) Strict flow matching: A valid frame must use the same flow IDs in all comparators in the engine except the PTP and MPLS comparators.
- (2) A valid frame may match any enabled flow within each comparator. There is one HW restriction mentioned above for flow match mode i.e. ingress and egress for same engine ID must have same flow match. In other words there is no provision to configure strict flow match in ingress, but non-strict flow match for egress. Same restriction for OAM engine 2A and 2B and also for ingress and egress i.e. engine 2A and 2B both ingress and egress must have same flow match mode.

Enumerator

VTSS_PHY_TS_ENG_FLOW_MATCH_ANY	match any flow in comparators
VTSS_PHY_TS_ENG_FLOW_MATCH_STRICT	strict flow match

Definition at line 813 of file vtss_phy_ts_api.h.

8.32.4.6 vtss_phy_ts_ptp_clock_mode_t

enum [vtss_phy_ts_ptp_clock_mode_t](#)

PTP Timestamp Engine operational modes.

Note

From the operational mode (vtss_phy_ts_ptp_clock_mode_t) and delay measurement method (vtss_phy_ts_ptp_delaym_type_t) the API sets up flows in the PTP comparator.

Enumerator

VTSS_PHY_TS_PTP_CLOCK_MODE_BC1STEP	Ordinary/Boundary clock, 1 step
VTSS_PHY_TS_PTP_CLOCK_MODE_BC2STEP	Ordinary/Boundary clock, 2 step
VTSS_PHY_TS_PTP_CLOCK_MODE_TC1STEP	Transparent clock, 1 step
VTSS_PHY_TS_PTP_CLOCK_MODE_TC2STEP	Transparent clock, 2 step
VTSS_PHY_TS_PTP_DELAY_COMP_ENGINE	Delay Compensation

Definition at line 1288 of file vtss_phy_ts_api.h.

8.32.4.7 vtss_phy_ts_ptp_delaym_type_t

enum [vtss_phy_ts_ptp_delaym_type_t](#)

PTP delay measurement method.

Note

As described above, using clock mode and delay measurement method, API sets up flows in PTP comparator.

Enumerator

VTSS_PHY_TS_PTP_DELAYM_P2P	Peer-to-Peer delay measurement method
VTSS_PHY_TS_PTP_DELAYM_E2E	End-to-End delay measurement method

Definition at line 1301 of file vtss_phy_ts_api.h.

8.32.4.8 vtss_phy_ts_y1731_oam_delaym_type_t

enum [vtss_phy_ts_y1731_oam_delaym_type_t](#)

Y.1731 OAM delay measurement method.

Note

Using delay measurement method, API sets up OAM flows in OAM comparator.

Enumerator

VTSS_PHY_TS_Y1731_OAM_DELAYM_1DM	One-Way delay measurement method
VTSS_PHY_TS_Y1731_OAM_DELAYM_DMM	Two-Way delay measurement method

Definition at line 1324 of file vtss_phy_ts_api.h.

8.32.4.9 vtss_phy_ts_ietf_mpls_ach_oam_delaym_type_t

enum [vtss_phy_ts_ietf_mpls_ach_oam_delaym_type_t](#)

IETF MPLS ACH, OAM delay measurement method.

Note

Using delay measurement method, API sets up OAM flows in OAM comparator.

Enumerator

VTSS_PHY_TS_IETF_MPLS_ACH_OAM_DELAY↔_M_DMM	Two-way delay measurement method
VTSS_PHY_TS_IETF_MPLS_ACH_OAM_DELAY↔_M_LDM	Loss/Delay Message combined Measurement Message

Definition at line 1334 of file vtss_phy_ts_api.h.

8.32.4.10 vtss_phy_ts_ietf_mpls_ach_oam_ts_format_t

enum [vtss_phy_ts_ietf_mpls_ach_oam_ts_format_t](#)

Analyzer OAM comparator, IETF MPLS ACH OAM Supported Timestamp Format.

Note

PTP Timestamp Format is Supported.

Enumerator

VTSS_PHY_TS_IETF_MPLS_ACH_OAM_TS_FORMAT_PTP	PTP - TimeStamp Format
---------------------------------------------	------------------------

Definition at line 1361 of file vtss_phy_ts_api.h.

8.32.4.11 vtss_phy_ts_action_format

enum [vtss_phy_ts_action_format](#)

Timestamp format to be configured in action configuration.

Enumerator

VTSS_PHY_TS_4_BYTE	Nano second timestamp
VTSS_PHY_TS_10_BYTE	10 byte timestamp

Definition at line 1392 of file vtss_phy_ts_api.h.

8.32.4.12 vtss_phy_ts_clockfreq_t

enum [vtss_phy_ts_clockfreq_t](#)

Timestamp block clock frequencies.

Enumerator

VTSS_PHY_TS_CLOCK_FREQ_125M	125 MHz
VTSS_PHY_TS_CLOCK_FREQ_15625M	156.25 MHz
VTSS_PHY_TS_CLOCK_FREQ_200M	200 MHz
VTSS_PHY_TS_CLOCK_FREQ_250M	250 MHz
VTSS_PHY_TS_CLOCK_FREQ_500M	500 MHz
VTSS_PHY_TS_CLOCK_FREQ_MAX	MAX Freq

Definition at line 1644 of file vtss_phy_ts_api.h.

8.32.4.13 vtss_phy_ts_clock_src_t

enum [vtss_phy_ts_clock_src_t](#)

Clock input source.

Enumerator

VTSS_PHY_TS_CLOCK_SRC_EXTERNAL	External source
VTSS_PHY_TS_CLOCK_SRC_CLIENT_RX	10G: XAUI lane 0 recovered clock, 1G: MAC RX clock (note: direction is opposite to 10G, i.e. PHY->MAC)
VTSS_PHY_TS_CLOCK_SRC_CLIENT_TX	10G: XAUI lane 0 recovered clock, 1G: MAC TX clock (note: direction is opposite to 10G, i.e. MAC->PHY)
VTSS_PHY_TS_CLOCK_SRC_LINE_RX	Received line clock
VTSS_PHY_TS_CLOCK_SRC_LINE_TX	transmitted line clock
VTSS_PHY_TS_CLOCK_SRC_INTERNAL	10G: Invalid, 1G: Internal 250 MHz Clock

Definition at line 1656 of file vtss_phy_ts_api.h.

8.32.4.14 vtss_phy_ts_rxtimestamp_pos_t

enum `vtss_phy_ts_rxtimestamp_pos_t`

defines Rx Timestamp position inside PTP frame.

Note

There are two options to put Rx timestamp in PTP frame: (a) Rx timestamp in Reserved 4 bytes of PTP header. (b) Shrink Preamble by 4 bytes and append 4 bytes at the end of frame. In this case Ethernet C↔RC will be overwritten by Rx timestamp and a new CRC will be appended after timestamp. Also note that Rx Timestamp position must be same for all the ports in the system; otherwise in ingress timestamp will be put in one position based on that port config whereas egress extract the time from different position as per that port config.

Enumerator

VTSS_PHY_TS_RX_TIMESTAMP_POS_IN_PTP	4 reserved bytes in PTP header
VTSS_PHY_TS_RX_TIMESTAMP_POS_AT_END	4 bytes appended at the end

Definition at line 1680 of file vtss_phy_ts_api.h.

8.32.4.15 vtss_phy_ts_rxtimestamp_len_t

enum `vtss_phy_ts_rxtimestamp_len_t`

Defines RX Timestamp format i.e. 30bit or 32bit Rx timestamp.

Note

30bit mode: The value in the reserved field is simply the nanosecCounter i.e. [0..999999999] 32bit mode: The value in the reserved field is a 32 bit value and equals: (nanosecCounter + secCounter*10^9) mod 2^32

Enumerator

VTSS_PHY_TS_RX_TIMESTAMP_LEN_30BIT	30 bit Rx timestamp
VTSS_PHY_TS_RX_TIMESTAMP_LEN_32BIT	32 bit Rx timestamp

Definition at line 1692 of file vtss_phy_ts_api.h.

8.32.4.16 `vtss_phy_ts_fifo_mode_t`

enum `vtss_phy_ts_fifo_mode_t`

Defines Tx TSFIFO access mode.

Enumerator

VTSS_PHY_TS_FIFO_MODE_NORMAL	in this mode, timestamp can be read from normal CPU interface
VTSS_PHY_TS_FIFO_MODE_SPI	Timestamps are pushed out on the SPI interface

Definition at line 1700 of file vtss_phy_ts_api.h.

8.32.4.17 `vtss_phy_ts_fifo_timestamp_len_t`

enum `vtss_phy_ts_fifo_timestamp_len_t`

Defines 4 bytes vs. the default 10 bytes Timestamp stored in Tx TSFIFO.

Enumerator

VTSS_PHY_TS_FIFO_TIMESTAMP_LEN_4BYTE	4 byte Tx timestamp
VTSS_PHY_TS_FIFO_TIMESTAMP_LEN_10BYTE	10 byte Tx timestamp

Definition at line 1708 of file vtss_phy_ts_api.h.

8.32.4.18 `vtss_phy_ts_tc_op_mode_t`

enum `vtss_phy_ts_tc_op_mode_t`

defines the Transparent Clock Operating Mode.

Note

There are two Modes TC can work: (a) Mode A: called SUB and ADD Mode where the local time is subtracted from the correction field at ingress and added at egress port. (b) Mode B: also called SUB_ADD Mode which uses reserve bytes (or append at the end of frame by replacing CRC) in PTP header to write RX_timestamp. (c) Mode C: also called 48-bit Mode. This mode uses 48-bits of the CF. This mode is similar to Mode A. At ingress local time is subtracted from CF and local time is added at egress. Also note that TC operating Mode must be same for all the ports in the system.

Enumerator

VTSS_PHY_TS_TC_OP_MODE_A	Sub local time at ingress and add at egress from CF
VTSS_PHY_TS_TC_OP_MODE_B	RX_timestamp using reserved bytes or append at the end as defined in vtss_phy_ts_rxtimestamp_pos_t
VTSS_PHY_TS_TC_OP_MODE_C	Sub local time at ingress and add at egress from CF and use 48 bits in CF

Definition at line 1727 of file vtss_phy_ts_api.h.

8.32.4.19 vtss_phy_ts_nphase_sampler_t

enum [vtss_phy_ts_nphase_sampler_t](#)

enum for n-phase samplers

Enumerator

VTSS_PHY_TS_NPHASE_PPS_O	N-Phase sampler for PPS_O
VTSS_PHY_TS_NPHASE_PPS_RI	N-Phase sampler for PPS_RI
VTSS_PHY_TS_NPHASE_EGR_SOF	N-Phase sampler for egress SOF
VTSS_PHY_TS_NPHASE_ING_SOF	N-Phase sampler for ingress SOF
VTSS_PHY_TS_NPHASE_LS	N-Phase sampler for Load/Save
VTSS_PHY_TS_NPHASE_MAX	Max N-Phase samplers

Definition at line 1873 of file vtss_phy_ts_api.h.

8.32.4.20 vtss_phy_ts_ptp_message_type_t

enum [vtss_phy_ts_ptp_message_type_t](#)

PTP Event Message TYPES.

Note

4 Types of Event messages.

Definition at line 2229 of file vtss_phy_ts_api.h.

8.32.5 Function Documentation

8.32.5.1 vtss_phy_ts_alt_clock_saved_get()

```
vtss_rc vtss_phy_ts_alt_clock_saved_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    u32 *const saved )
```

Get the latest saved nanosec counter from the alternative clock.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>saved</i>	[OUT] latest saved value.

Returns

Return code.

8.32.5.2 vtss_phy_ts_alt_clock_mode_get()

```
vtss_rc vtss_phy_ts_alt_clock_mode_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_ts_alt_clock_mode_t *const phy_alt_clock_mode )
```

Get the alternative external clock mode.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>phy_alt_clock_mode</i>	[OUT] alternative clock mode.

Returns

Return code.

8.32.5.3 vtss_phy_ts_alt_clock_mode_set()

```
vtss_rc vtss_phy_ts_alt_clock_mode_set (
    const vtss_inst_t inst,
```

```
const vtss_port_no_t port_no,
const vtss_phy_ts_alt_clock_mode_t *const phy_alt_clock_mode )
```

Set the alternative clock mode. This function configures the loopbacks.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>phy_alt_clock_mode</i>	[IN] alternative clock mode.

Returns

Return code.

8.32.5.4 vtss_phy_ts_pps_conf_set()

```
vtss_rc vtss_phy_ts_pps_conf_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_ts_pps_conf_t *const phy_pps_conf )
```

Set offset for the PPS generation.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>phy_pps_conf</i>	[IN] pps configuration

Returns

Return code.

8.32.5.5 vtss_phy_ts_pps_conf_get()

```
vtss_rc vtss_phy_ts_pps_conf_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_ts_pps_conf_t *const phy_pps_conf )
```

Get offset for the PPS generation.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>phy_pps_conf</i>	[OUT] pps configuration

Generated by Doxygen

Returns

Return code.

8.32.5.6 vtss_phy_ts_ingress_latency_set()

```
vtss_rc vtss_phy_ts_ingress_latency_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_timeinterval_t *const latency )
```

Set the ingress latency.

Note

Latency is taken as TimeInterval which is 48bit nanosec + 16bit sub-nanosec though HW supports nanosec in the range 0 - 2^{16} .

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>latency</i>	[IN] ingress latency

Returns

Return code.

8.32.5.7 vtss_phy_ts_ingress_latency_get()

```
vtss_rc vtss_phy_ts_ingress_latency_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_timeinterval_t *const latency )
```

Get the ingress latency.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>latency</i>	[OUT] ingress latency

Returns

Return code.

8.32.5.8 vtss_phy_ts_egress_latency_set()

```
vtss_rc vtss_phy_ts_egress_latency_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_timeinterval_t *const latency )
```

Set the egress latency.

Note

Latency is taken as TimeInterval which is 48bit nanosec + 16bit sub-nanosec though HW supports nanosec in the range 0 - 2^{16} .

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>latency</i>	[IN] egress latency

Returns

Return code.

8.32.5.9 vtss_phy_ts_egress_latency_get()

```
vtss_rc vtss_phy_ts_egress_latency_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_timeinterval_t *const latency )
```

Get the egress latency.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>latency</i>	[OUT] egress latency

Returns

Return code.

8.32.5.10 vtss_phy_ts_path_delay_set()

```
vtss_rc vtss_phy_ts_path_delay_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_timeinterval_t *const path_delay )
```

Set the path delay.

Note

Path delay is taken as TimeInterval which is 48bit nanosec + 16bit sub-nanosec though HW supports nanosec in the range 0 - 2^{32} .

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>path_delay</i>	[IN] path delay (measured)

Returns

Return code.

8.32.5.11 vtss_phy_ts_path_delay_get()

```
vtss_rc vtss_phy_ts_path_delay_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_timeinterval_t *const path_delay )
```

Get the path delay.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>path_delay</i>	[OUT] path delay

Returns

Return code.

8.32.5.12 vtss_phy_ts_delay_asymmetry_set()

```
vtss_rc vtss_phy_ts_delay_asymmetry_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_timeinterval_t *const delay_asym )
```

Set the delay asymmetry.

Note

Asymmetry is taken as TimeInterval which is 48bit nanosec + 16bit sub-nanosec though HW supports scaled nanosec which is 16 bit nanosec + 16 bit sub-nanosec, i.e. the range is $-2^{15} - (+2^{15}-2^{-16})$.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>delay_asym</i>	[IN] link delay asymmetry

Returns

Return code.

8.32.5.13 vtss_phy_ts_delay_asymmetry_get()

```
vtss_rc vtss_phy_ts_delay_asymmetry_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_timeinterval_t *const delay_asym )
```

Get the delay asymmetry.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>delay_asym</i>	[OUT] link delay asymmetry

Returns

Return code.

8.32.5.14 vtss_phy_ts_ptptime_set()

```
vtss_rc vtss_phy_ts_ptptime_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_timestamp_t *const ts )
```

Set the current PTP time into the PHY.

Note

Time to be set must be next pps time.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>ts</i>	[IN] current PTP time

Returns

Return code.

8.32.5.15 vtss_phy_ts_ptptime_set_done()

```
vtss_rc vtss_phy_ts_ptptime_set_done (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no )
```

Setting of the current PTP time into the PHY is completed.

Note

This function should be called after the next pps after setting the next pps time.

Parameters

<i>inst</i>	[IN] handle to an API instance.
<i>port_no</i>	[IN] port number

Returns

Return code.

8.32.5.16 vtss_phy_ts_ptptime_arm()

```
vtss_rc vtss_phy_ts_ptptime_arm (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no )
```

Arm the local time of the PHY so that in next pps it can be read.

Note

Once armed, in next pps it will load the local time and can be read using vtss_phy_ts_ptptime_get. Must call before get the local time of the PHY.

Parameters

<i>inst</i>	[IN] handle to an API instance.
<i>port_no</i>	[IN] port number

Returns

Return code.

8.32.5.17 vtss_phy_ts_ptptime_get()

```
vtss_rc vtss_phy_ts_ptptime_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_timestamp_t *const ts )
```

Get the armed PTP time from the PHY.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>ts</i>	[OUT] local time in PHY

Returns

Return code. If the time has not been updated after the vtss_phy_ts_ptptime_arm function is called, it returns error.

8.32.5.18 vtss_phy_ts_load_ptptime_get()

```
vtss_rc vtss_phy_ts_load_ptptime_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_timestamp_t *const ts )
```

Get the PTP time from the PHY load registers.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>ts</i>	[OUT] local time in PHY

Returns

Return code. If the time has not been updated after the vtss_phy_ts_ptptime_arm function is called, it returns error.

8.32.5.19 vtss_phy_ts_sertod_set()

```
vtss_rc vtss_phy_ts_sertod_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_ts_sertod_conf_t *const sertod_conf )
```

Set Enable/Disable Serial ToD.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>sertod_conf</i>	[IN] configure Serial ToD input

Returns

Return code.

8.32.5.20 vtss_phy_ts_sertod_get()

```
vtss_rc vtss_phy_ts_sertod_get (
    const vtss_inst_t inst,
```

```
const vtss_port_no_t port_no,
vtss_phy_ts_sertod_conf_t *const sertod_conf )
```

Get Enable/Disable Serial ToD.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>sertod_conf</i>	[OUT] Serial ToD configuration

Returns

Return code.

8.32.5.21 vtss_phy_ts_loadpulse_delay_set()

```
vtss_rc vtss_phy_ts_loadpulse_delay_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const u16 *const delay )
```

Set load pulse delay.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>delay</i>	[IN] delay value in nano seconds

Returns

Return code.

8.32.5.22 vtss_phy_ts_loadpulse_delay_get()

```
vtss_rc vtss_phy_ts_loadpulse_delay_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    u16 *const delay )
```

Get load pulse delay.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>delay</i>	[OUT] delay value in nano seconds

Returns

Return code.

8.32.5.23 vtss_phy_ts_clock_rateadj_set()

```
vtss_rc vtss_phy_ts_clock_rateadj_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_ts_scaled_ppb_t *const adj )
```

Adjust the local clock rate.

Parameters

<i>inst</i>	[IN] handle to an API instance.
<i>port_no</i>	[IN] port number
<i>adj</i>	[IN] Clock ratio frequency offset in units of scaled ppb (parts pr billion). ratio > 0 => clock runs faster.

Returns

Return code.

8.32.5.24 vtss_phy_ts_clock_rateadj_get()

```
vtss_rc vtss_phy_ts_clock_rateadj_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_ts_scaled_ppb_t *const adj )
```

Get the clock rate adjustment value.

Parameters

<i>inst</i>	[IN] handle to an API instance.
<i>port_no</i>	[IN] port number
<i>adj</i>	[OUT] Clock ratio frequency offset in scaled ppb. ratio > 0 => clock runs faster

Returns

Return code.

8.32.5.25 vtss_phy_ts_clock_rateadj_ppm_set()

```
vtss_rc vtss_phy_ts_clock_rateadj_ppm_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_ts_scaled_ppb_t *const adj )
```

Adjust ppm of the local clock rate .

Parameters

<i>inst</i>	[IN] handle to an API instance.
<i>port_no</i>	[IN] port number
<i>adj</i>	[IN] Clock ratio frequency offset in units of scaled ppb (parts per billion). ratio > 0 => clock runs faster.

Returns

Return code.

8.32.5.26 vtss_phy_ts_clock_rateadj_ppm_get()

```
vtss_rc vtss_phy_ts_clock_rateadj_ppm_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_ts_scaled_ppb_t *const adj )
```

Get the clock rate ppm adjustment value.

Parameters

<i>inst</i>	[IN] handle to an API instance.
<i>port_no</i>	[IN] port number
<i>adj</i>	[OUT] Clock ratio frequency offset in scaled ppb. ratio > 0 => clock runs faster

Returns

Return code.

8.32.5.27 vtss_phy_ts_ptptime_adj1ns()

```
vtss_rc vtss_phy_ts_ptptime_adj1ns (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const BOOL incr )
```

Increment/decrement the LTC clock value by 1 ns.

Parameters

<i>inst</i>	[IN] handle to an API instance.
<i>port_no</i>	[IN] port number
<i>incr</i>	[IN] increment/decrement: TRUE = incr, FALSE = decr

Returns

Return code.

8.32.5.28 vtss_phy_ts_timeofday_offset_set()

```
vtss_rc vtss_phy_ts_timeofday_offset_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const i32 offset )
```

Subtract offset from the current time.

Parameters

<i>inst</i>	[IN] handle to an API instance.
<i>port_no</i>	[IN] port number
<i>offset</i>	[IN] offset in nano seconds

Returns

Return code.

8.32.5.29 vtss_phy_ts_ongoing_adjustment()

```
vtss_rc vtss_phy_ts_ongoing_adjustment (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_ts_todadj_status_t *const ongoing_adjustment )
```

Return the status of the LTC time adjustment.

Parameters

<i>inst</i>	[IN] handle to an API instance.
<i>port_no</i>	[IN] port number
<i>ongoing_adjustment</i>	[OUT] LTC offset operation status

Returns

Return code.

8.32.5.30 vtss_phy_ts_ltc_freq_synth_pulse_set()

```
vtss_rc vtss_phy_ts_ltc_freq_synth_pulse_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_ltc_freq_synth_t *const ltc_freq_synthesis )
```

Return the status of the LTC time adjustment.

Parameters

<i>inst</i>	[IN] handle to an API instance.
<i>port_no</i>	[IN] port number
<i>ltc_freq_synthesis</i>	[IN] Frequency synthesis pulse configuration

Returns

Return code.

8.32.5.31 vtss_phy_ts_ltc_freq_synth_pulse_get()

```
vtss_rc vtss_phy_ts_ltc_freq_synth_pulse_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_ltc_freq_synth_t *const ltc_freq_synthesis )
```

Return the status of the LTC time adjustment.

Parameters

<i>inst</i>	[IN] handle to an API instance.
<i>port_no</i>	[IN] port number
<i>ltc_freq_synthesis</i>	[OUT] Frequency synthesis pulse configuration

Returns

Return code.

8.32.5.32 vtss_phy_daisy_conf_set()

```
vtss_rc vtss_phy_daisy_conf_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_daisy_chain_conf_t *const daisy_chain )
```

configure the daisy chain for TS FIFO

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>daisy_chain</i>	[IN] daisy-chaining configuration

Returns

Return code.

8.32.5.33 vtss_phy_daisy_conf_get()

```
vtss_rc vtss_phy_daisy_conf_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_daisy_chain_conf_t *const daisy_chain )
```

getting the daisy chain for TS FIFO

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>daisy_chain</i>	[IN] daisy-chaining configuration

Returns

Return code.

8.32.5.34 vtss_phy_ts_fifo_sig_set()

```
vtss_rc vtss_phy_ts_fifo_sig_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_ts_fifo_sig_mask_t sig_mask )
```

Set Tx TSFIFO i.e. frame signature in Tx TSFIFO.

Note

Ports sharing timestamp IP block use common register in analyzer to configure the signature i.e. all the ports within the timestamp IP block will have same signature in TSFIFO. In other words, to configure the signature in the FIFO, any of the ports within timestamp block can be used.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>sig_mask</i>	[IN] signature mask

Returns

Return code.

8.32.5.35 vtss_phy_ts_fifo_sig_get()

```
vtss_rc vtss_phy_ts_fifo_sig_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_ts_fifo_sig_mask_t *const sig_mask )
```

Get frame signature mask in Tx TSFIFO.

Note

As described in vtss_phy_ts_fifo_sig_set, any of the ports within IP timestamp block can be used to get the signature.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>sig_mask</i>	[OUT] signature mask

Returns

Return code.

8.32.5.36 vtss_phy_ts_fifo_empty()

```
vtss_rc vtss_phy_ts_fifo_empty (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no )
```

Read timestamp from Tx TSFIFO.

Note

Application will call this function upon receipt of a signal for timestamp in FIFO.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number

Returns

Return code.

8.32.5.37 vtss_phy_ts_fifo_read_install()

```
vtss_rc vtss_phy_ts_fifo_read_install (
    const vtss_inst_t inst,
    vtss_phy_ts_fifo_read rd_cb,
    void * ctxt )
```

Install callback to read data (signature + timestamp) from Tx TSFIFO.

Note

Registered callback will be called for each entry in TSFIFO from vtss_phy_ts_fifo_empty function.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>rd_cb</i>	[IN] read callback
<i>ctxt</i>	[IN] context to be returned in callback

Returns

Return code.

8.32.5.38 vtss_phy_ts_fifo_read_cb_get()

```
vtss_rc vtss_phy_ts_fifo_read_cb_get (
    const vtss_inst_t inst,
    vtss_phy_ts_fifo_read * rd_cb,
    void ** ctxt )
```

Get the fifo read callback function installed.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>rd_cb</i>	[OUT] read callback
<i>ctxt</i>	[OUT] context

Returns

Return code.

8.32.5.39 vtss_phy_ts_ingress_engine_init()

```
vtss_rc vtss_phy_ts_ingress_engine_init (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_ts_engine_t eng_id,
    const vtss_phy_ts_encap_t encapsulation,
    const u8 flow_st_index,
    const u8 flow_end_index,
    const vtss_phy_ts_engine_flow_match_t flow_match_mode )
```

Initialize an analyzer ingress engine for an encapsulation type.

Note

Depending on chip type, an analyzer engine may be shared by two ports, and therefore no need to initialize separate engine for each port. If the same encapsulation is used by the two ports, both can use same engine. To initialize an engine shared by both ports, either of the ports no can be passed to this API and use flow_map within the engine to map flows to the ports.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>eng_id</i>	[IN] engine ID
<i>encapsulation</i>	[IN] frame encapsulation
<i>flow_st_index</i>	[IN] flow start index
<i>flow_end_index</i>	[IN] flow end index
<i>flow_match_mode</i>	[IN] flow match mode: strict/non-strict flow match

Returns

Return code.

8.32.5.40 vtss_phy_ts_ingress_engine_init_conf_get()

```
vtss_rc vtss_phy_ts_ingress_engine_init_conf_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_ts_engine_t eng_id,
    vtss_phy_ts_eng_init_conf_t *const init_conf )
```

Get the configuration parameters passed in engine_init of an analyzer ingress engine for a specific engine ID.

Note

Depending on chip type, an analyzer engine may be shared by two ports, and therefore the configuration is same for both the ports.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>eng_id</i>	[IN] engine ID
<i>init_conf</i>	[OUT] config parameters in engine_init

Returns

Return code.

8.32.5.41 vtss_phy_ts_ingress_engine_clear()

```
vtss_rc vtss_phy_ts_ingress_engine_clear (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_ts_engine_t eng_id )
```

Clear/release an analyzer ingress engine already initialized.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>eng_id</i>	[IN] engine ID

Returns

Return code.

8.32.5.42 vtss_phy_ts_egress_engine_init()

```
vtss_rc vtss_phy_ts_egress_engine_init (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_ts_engine_t eng_id,
    const vtss_phy_ts_encap_t encapsulation,
    const u8 flow_st_index,
    const u8 flow_end_index,
    const vtss_phy_ts_engine_flow_match_t flow_match_mode )
```

Initialize an analyzer egress engine for an encapsulation type.

Note

Depending on chip type, an analyzer engine may be shared by two ports, and therefore no need to initialize separate engine for each port. If the same encapsulation is used by the two ports, both can use same engine. To initialize an engine shared by both ports, either of the ports no can be passed to this API and use flow_map within the engine to map flows to the ports.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>eng_id</i>	[IN] engine ID
<i>encapsulation</i>	[IN] frame encapsulation
<i>flow_st_index</i>	[IN] flow start index
<i>flow_end_index</i>	[IN] flow end index
<i>flow_match_mode</i>	[IN] flow match mode: strict/non-strict flow match

Returns

Return code.

8.32.5.43 vtss_phy_ts_egress_engine_init_conf_get()

```
vtss_rc vtss_phy_ts_egress_engine_init_conf_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_ts_engine_t eng_id,
    vtss_phy_ts_eng_init_conf_t *const init_conf )
```

Get the configuration parameters passed in engine_init of an analyzer egress engine for a specific engine ID.

Note

Depending on chip type, an analyzer engine may be shared by two ports, and therefore the configuration is same for both the ports.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>eng_id</i>	[IN] engine ID
<i>init_conf</i>	[OUT] config parameters in engine_init

Returns

Return code.

8.32.5.44 vtss_phy_ts_egress_engine_clear()

```
vtss_rc vtss_phy_ts_egress_engine_clear (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_ts_engine_t eng_id )
```

Clear/release an analyzer egress engine already initialized.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>eng_id</i>	[IN] engine ID

Returns

Return code.

8.32.5.45 vtss_phy_ts_ingress_engine_conf_set()

```
vtss_rc vtss_phy_ts_ingress_engine_conf_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_ts_engine_t eng_id,
    const vtss_phy_ts_engine_flow_conf_t *const flow_conf )
```

Configure ingress analyzer flow.

Note

For multi-port timestamp block, either of the port can be used to configure the ingress engine, port to which frame will be matched is specified in flow-map parameter. Note that during the process of updating an engine configuration, the engine mode will be disabled and it will start processing PTP frame after config finished. Also if the local time counter is out of sync, and has to be set, then the received ptp packets must be ignored, and no ptp packets should be transmitted, but this should be controlled by the application.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>eng_id</i>	[IN] engine ID
<i>flow_conf</i>	[IN] pointer to engine configuration

Returns

Return code.

8.32.5.46 vtss_phy_ts_ingress_engine_conf_get()

```
vtss_rc vtss_phy_ts_ingress_engine_conf_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_ts_engine_t eng_id,
    vtss_phy_ts_engine_flow_conf_t *const flow_conf )
```

Get ingress analyzer flow.

Note

For multi-port timestamp block, either of the port can used to get the ingress engine configuration.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>eng_id</i>	[IN] engine ID
<i>flow_conf</i>	[OUT] pointer to engine configuration

Returns

Return code.

8.32.5.47 vtss_phy_ts_egress_engine_conf_set()

```
vtss_rc vtss_phy_ts_egress_engine_conf_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_ts_engine_t eng_id,
    const vtss_phy_ts_engine_flow_conf_t *const flow_conf )
```

Configure egress analyzer flow.

Note

For multi-port timestamp block, either of the port can be used to configure the ingress engine, port to which frame will be matched is specified in flow_map parameter.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>eng_id</i>	[IN] engine ID
<i>flow_conf</i>	[IN] pointer to engine configuration

Returns

Return code.

8.32.5.48 vtss_phy_ts_egress_engine_conf_get()

```
vtss_rc vtss_phy_ts_egress_engine_conf_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_ts_engine_t eng_id,
    vtss_phy_ts_engine_flow_conf_t *const flow_conf )
```

Get egress analyzer flow.

Note

For multi-port timestamp block, either of the port can be used to get the ingress engine configuration.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>eng_id</i>	[IN] engine ID
<i>flow_conf</i>	[OUT] pointer to engine configuration

Returns

Return code.

8.32.5.49 vtss_phy_ts_ingress_engine_action_set()

```
vtss_rc vtss_phy_ts_ingress_engine_action_set (
    const vtss_inst_t inst,
```

```
const vtss_port_no_t port_no,
const vtss_phy_ts_engine_t eng_id,
const vtss_phy_ts_engine_action_t *const action_conf )
```

Configure ingress analyzer engine action.

Note

For multi-port timestamp block, either of the port can be used to configure the ingress engine. Note that during the process of updating an engine configuration, the engine mode will be disabled and it will start processing frame after config finished.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>eng_id</i>	[IN] engine ID
<i>action_conf</i>	[IN] action associated with the engine

Returns

Return code.

8.32.5.50 vtss_phy_ts_ingress_engine_action_get()

```
vtss_rc vtss_phy_ts_ingress_engine_action_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_ts_engine_t eng_id,
    vtss_phy_ts_engine_action_t *const action_conf )
```

Get ingress analyzer engine action. For multi-port timestamp block, either of the port can be used to get the ingress engine.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>eng_id</i>	[IN] engine ID
<i>action_conf</i>	[OUT] action associated with the engine

Returns

Return code.

8.32.5.51 vtss_phy_ts_egress_engine_action_set()

```
vtss_rc vtss_phy_ts_egress_engine_action_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_ts_engine_t eng_id,
    const vtss_phy_ts_engine_action_t *const action_conf )
```

Configure egress analyzer engine action.

Note

For multi-port timestamp block, either of the port can be used to configure the ingress engine. Note that during the process of updating an engine configuration, the engine mode will be disabled and it will start processing frame after config finished.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>eng_id</i>	[IN] engine ID
<i>action_conf</i>	[IN] action associated with the engine

Returns

Return code.

8.32.5.52 vtss_phy_ts_egress_engine_action_get()

```
vtss_rc vtss_phy_ts_egress_engine_action_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_ts_engine_t eng_id,
    vtss_phy_ts_engine_action_t *const action_conf )
```

Get egress analyzer engine action. For multi-port timestamp block, either of the port can be used to get the ingress engine.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>eng_id</i>	[IN] engine ID
<i>action_conf</i>	[OUT] action associated with the engine

Returns

Return code.

8.32.5.53 vtss_phy_ts_event_enable_set()

```
vtss_rc vtss_phy_ts_event_enable_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const BOOL enable,
    const vtss_phy_ts_event_t ev_mask )
```

Enabling / Disabling of events.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>enable</i>	[IN] Enable or disable events
<i>ev_mask</i>	[IN] Event type(s) to control (mask)

Returns

Return code.

8.32.5.54 vtss_phy_ts_event_enable_get()

```
vtss_rc vtss_phy_ts_event_enable_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_ts_event_t *const ev_mask )
```

Get Enabling of events.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>ev_mask</i>	[OUT] Mask containing events that are enabled

Returns

Return code.

8.32.5.55 vtss_phy_ts_event_poll()

```
vtss_rc vtss_phy_ts_event_poll (
    const vtss_inst_t inst,
```

```
const vtss_port_no_t port_no,
vtss_phy_ts_event_t *const status )
```

Polling function called at by interrupt or periodically.

Note

Interrupt status will be cleared on read

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>status</i>	[OUT] Event status, bit set indicates corresponding event/interrupt has detected

Returns

Return code.

8.32.5.56 vtss_phy_ts_stats_get()

```
vtss_rc vtss_phy_ts_stats_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_ts_stats_t *const statistics )
```

Get Timestamp statistics.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>statistics</i>	[OUT] pointer to statistics structure

Returns

Return code.

8.32.5.57 vtss_phy_ts_correction_overflow_get()

```
vtss_rc vtss_phy_ts_correction_overflow_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    BOOL *const ingr_overflow,
    BOOL *const egr_overflow )
```

Get the correction field overflow status in ingress and egress.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>ingr_overflow</i>	[OUT] ingress overflow status (enable/disable)
<i>egr_overflow</i>	[OUT] egress overflow status (enable/disable)

Returns

Return code.

8.32.5.58 vtss_phy_ts_mode_set()

```
vtss_rc vtss_phy_ts_mode_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const BOOL enable )
```

Enable/disable timestamp block.

Note

Disabling the timestamp block will 'BYPASS' the block.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>enable</i>	[IN] enable/disable parameter

Returns

Return code.

8.32.5.59 vtss_phy_ts_mode_get()

```
vtss_rc vtss_phy_ts_mode_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    BOOL *const enable )
```

Get timestamp block status i.e. enable/disable.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>enable</i>	[OUT] enable/disable parameter

Returns

Return code.

8.32.5.60 vtss_phy_ts_init()

```
vtss_rc vtss_phy_ts_init (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_ts_init_conf_t *const conf )
```

Init timestamp block.

Note

Init has to be called for each port in the time stamp block.

Parameters

<i>inst</i>	[IN] Target instance reference
<i>port_no</i>	[IN] Port number
<i>conf</i>	[IN] Init config parameters

Returns

Return code.

8.32.5.61 vtss_phy_ts_init_conf_get()

```
vtss_rc vtss_phy_ts_init_conf_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    BOOL *const init_done,
    vtss_phy_ts_init_conf_t *const conf )
```

Get the timestamp init config parameters.

Parameters

<i>inst</i>	[IN] Target instance reference
<i>port_no</i>	[IN] Port number
<i>init_done</i>	[OUT] Timestamp init done or not for the port
<i>conf</i>	[OUT] Init config parameters

Returns

Return code.

8.32.5.62 vtss_phy_ts_nphase_status_get()

```
vtss_rc vtss_phy_ts_nphase_status_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_ts_nphase_sampler_t sampler,
    vtss_phy_ts_nphase_status_t *const status )
```

Get N-Phase sampler status.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>sampler</i>	[IN] N-Phase sampler type
<i>status</i>	[OUT] status

Returns

Return code.

8.32.5.63 vtss_phy_ts_hiacc_set()

```
vtss_rc vtss_phy_ts_hiacc_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_ts_nphase_sampler_t sampler,
    const BOOL enable )
```

Enable N-Phase sampler.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>sampler</i>	[IN] N-Phase sampler type
<i>enable</i>	[IN] enable/disable N-Phase sampler

Returns

Return code.

8.32.5.64 vtss_phy_ts_hiacc_get()

```
vtss_rc vtss_phy_ts_hiacc_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_ts_nphase_sampler_t sampler,
    BOOL const * enable )
```

N-Phase sampler status get.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>sampler</i>	[IN] N-Phase sampler type
<i>enable</i>	[OUT] enable/disable N-Phase sampler

Returns

Return code.

8.32.5.65 vtss_phy_ts_block_soft_reset()

```
vtss_rc vtss_phy_ts_block_soft_reset (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_ts_soft_reset_t ts_reset )
```

reset 1588 block.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>ts_reset</i>	[IN] 1588 block reset

Returns

Return code.

8.32.5.66 vtss_phy_ts_new_spi_mode_set()

```
vtss_rc vtss_phy_ts_new_spi_mode_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const BOOL enable )
```

Enable/disable New SPI mode for 8574-15 (Rev A & Rev B) that uses PPS0 pin as the new SPI_CLK.

Note

This will activate the New SPI bus while enabled. User must make note of the following: (1) SPI mode to access TS_FIFO has to be mentioned in vtss_phy_ts_init. (2) Both the ports of a block must have same configuration in terms of FIFO access mode i.e. MDIO, Old SPI Mode or New SPI mode. Also Old and New SPI cannot be mixed between blocks of a chip and both the blocks must be in same SPI mode i.e. both must be either Old SPI or New SPI. (3) This must be the last step after all the config done for both the ports of the block. (4) This is required/supported for 8574-15 RevA and RevB, not for 8487/8488-15.

Parameters

<i>inst</i>	[IN] Target instance reference
<i>port_no</i>	[IN] Port number
<i>enable</i>	[IN] enable/disable of New SPI. If tx_fifo_mode is VTSS_PHY_TS_FIFO_MODE_SPI: disable => SPI is in Old SPI Mode enable => SPI in New SPI Mode

Returns

Return code.

8.32.5.67 vtss_phy_ts_new_spi_mode_get()

```
vtss_rc vtss_phy_ts_new_spi_mode_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    BOOL *const mode )
```

Get New SPI mode for 8574-15 (Rev A & Rev B) described above.

Parameters

<i>inst</i>	[IN] Target instance reference
<i>port_no</i>	[IN] Port number
<i>mode</i>	[OUT] Status of new SPI mode disable => SPI is in Old SPI Mode enable => SPI in New SPI Mode

Returns

Return code.

8.32.5.68 vtss_phy_ts_phy_oper_mode_change()

```
vtss_rc vtss_phy_ts_phy_oper_mode_change (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no )
```

Update the PHY timestamping block to predict the correct latency.

Note

This function should be called when changing the PHY oper mode. Eg:LAN to WAN The application must configure the Latency values corresponding to the active mode.

Parameters

<i>inst</i>	[IN] Target instance reference
<i>port_no</i>	[IN] Port number

Returns

Return code.

8.32.5.69 vtss_phy_1588_csr_reg_write()

```
vtss_rc vtss_phy_1588_csr_reg_write (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const u32 blk_id,
    const u16 csr_address,
    const u32 *const value )
```

Set the the 1588 block CSR registers.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>blk_id</i>	[IN] 1588 CSR block Index [0-7]
<i>csr_address</i>	[IN] 1588 CSR block Register Offset [0x00 - 0x7f]
<i>value</i>	[IN] 32 bit value

Returns

Return code.

8.32.5.70 vtss_phy_1588_csr_reg_read()

```
vtss_rc vtss_phy_1588_csr_reg_read (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const u32 blk_id,
    const u16 csr_address,
    u32 *const value )
```

get the the 1588 block CSR registers.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>blk_id</i>	[IN] 1588 CSR block Index [0-7]
<i>csr_address</i>	[IN] 1588 CSR block Register Offset [0x00 - 0x7f]
<i>value</i>	[OUT] 32 bit value

Returns

Return code.

8.32.5.71 vtss_phy_ts_status_check()

```
vtss_rc vtss_phy_ts_status_check (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    BOOL wait,
    const vtss_debug_printf_t pr )
```

TS status check function supported for 10G Phys like 8488 & 8492.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>wait</i>	[IN] wait if needed
<i>pr</i>	[IN] print function to be passed for default logging

Returns

Return code.

8.32.5.72 vtss_phy_ts_10g_extended_fifo_sync()

```
vtss_rc vtss_phy_ts_10g_extended_fifo_sync (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_10g_fifo_sync_t * conf )
```

Reset 1588 PHY Fifo oos. As of now, supported for 10G Phys like 8488.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>conf</i>	[IN] Select modes for running the API.

Returns

Return code.

8.32.5.73 vtss_phy_ts_10g_fifo_sync()

```
vtss_rc vtss_phy_ts_10g_fifo_sync (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_debug_printf_t pr )
```

Reset 1588 PHY Fifo oos. As of now, supported for 10G Phys like 8488.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>pr</i>	[IN] print function to be passed for default logging

Returns

Return code.

8.32.5.74 vtss_phy_ts_bypass_clear()

```
vtss_rc vtss_phy_ts_bypass_clear (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_debug_printf_t pr )
```

1588 Bypass clear

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>pr</i>	[IN] print function to be passed for default logging

Returns

Return code.

8.32.5.75 vtss_phy_ts_viper_fifo_reset()

```
vtss_rc vtss_phy_ts_viper_fifo_reset (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_phy_ts_fifo_conf_t * fifo_conf )
```

Viper 1588 FIFO reset.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>fifo_conf</i>	[IN] FIFO algorithm Operation mode.

Returns

Return Code.

8.32.5.76 vtss_phy_ts_tesla_tsp_fifo_sync()

```
vtss_rc vtss_phy_ts_tesla_tsp_fifo_sync (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_debug_printf_t pr,
    const vtss_phy_ts_fifo_conf_t * fifo_conf,
    BOOL * OOS )
```

API to detect and correct Timestamp FIFO OOS.

Note

Compile time flag TESLA_ING_TS_ERRFIX is needed for this API to work else this API will have no effect

Parameters

<i>inst</i>	[IN] Target instance reference
<i>port_no</i>	[IN] Port number
<i>pr</i>	[IN] print function needed for default logging
<i>fifo_conf</i>	[IN] fifo config struct
<i>OOS</i>	[OUT] True/False of Out-of-Sync for both Ingress and Egress combined

Returns

Return code.

8.32.5.77 vtss_phy_1g_ts_fifo_sync()

```
vtss_rc vtss_phy_1g_ts_fifo_sync (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_debug_printf_t pr,
    const vtss_phy_ts_fifo_conf_t * fifo_conf,
    BOOL * OOS )
```

API to detect and correct Timestamp FIFO OOS for 1G PHY's (Viper and Tesla)

Note

: In Viper OOS API there is no detection mechanism.

Parameters

<i>inst</i>	[IN] Target instance reference
<i>port_no</i>	[IN] Port number
<i>pr</i>	[IN] print function needed for default logging
<i>fifo_conf</i>	[IN] fifo config struct
<i>OOS</i>	[OUT] True/False of Out-of-Sync for both Ingress and Egress combined(Only for Tesla)

Returns

Return code.

8.32.5.78 vtss_phy_1588_debug_reg_read()

```
vtss_rc vtss_phy_1588_debug_reg_read (
    const vtss_inst_t inst,
```

```
const vtss_port_no_t port_no,
const u32 blk_id,
const vtss_debug_printf_t p_routine )
```

API to dump PHY timestamp registers (for Debugging)

Parameters

<i>inst</i>	[IN] Target instance reference
<i>port_no</i>	[IN] Port number
<i>blk_id</i>	[IN] Register block id
<i>p_routine</i>	[IN] print function needed for default logging

Returns

Return code.

8.32.5.79 vtss_phy_ts_flow_clear_cf_set()

```
vtss_rc vtss_phy_ts_flow_clear_cf_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    BOOL ingress,
    const vtss_phy_ts_engine_t eng_id,
    u8 act_id,
    vtss_phy_ts_ptp_message_type_t msgtype )
```

Clear Correction field for specified PTP message type.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>ingress</i>	[IN] TRUE if Ingress elses Egress
<i>eng_id</i>	[IN] 1588 Engine ID
<i>act_id</i>	[IN] 1588 Action ID
<i>msgtype</i>	[IN] PTP Message Type

Returns

Return code.

8.33 vtss_api/include/vtss_port_api.h File Reference

Port API.

```
#include <vtss/api/port.h>
```

Data Structures

- struct `vtss_port_map_t`
Port map structure.
- struct `vtss_port_clause_37_adv_t`
Advertisement control data for Clause 37 aneg.
- struct `vtss_port_sgmii_aneg_t`
Advertisement control data for SGMII aneg.
- struct `vtss_port_clause_37_control_t`
Auto-negotiation control parameter struct.
- struct `vtss_port_flow_control_conf_t`
Flow control setup.
- struct `vtss_port_frame_gaps_t`
Inter frame gap structure.
- struct `vtss_port_serdes_conf_t`
SFI Serdes configuration.
- struct `vtss_port_conf_t`
Port configuration structure.
- struct `vtss_basic_counters_t`
Basic counters structure.
- struct `vtss_port_ifh_t`
Port Internal Frame Header structure.

Macros

- `#define CHIP_PORT_UNUSED -1`
- `#define VTSS_FRAME_GAP_DEFAULT 0`
- `#define VTSS_MAX_FRAME_LENGTH_STANDARD 1518`
- `#define VTSS_MAX_FRAME_LENGTH_MAX 10240`
- `#define VTSS_MAX_FRAME_LENGTH_MAX 4776`

Enumerations

- enum `vtss_miim_controller_t` { `VTSS_MIIM_CONTROLLER_0` = 0, `VTSS_MIIM_CONTROLLER_1` = 1, `VTSS_MIIM_CONTROLLERS`, `VTSS_MIIM_CONTROLLER_NONE` = -1 }
MII management controller.
- enum `vtss_internal_bw_t` { `VTSS_BW_DEFAULT`, `VTSS_BW_1G`, `VTSS_BW_2500M`, `VTSS_BW_UNDEFINED` }
The internal bandwidth allocated for the port.
- enum `vtss_port_clause_37_remote_fault_t` { `VTSS_PORT_CLAUSE_37_RF_LINK_OK` = ((0<<1) | (0<<0)), `VTSS_PORT_CLAUSE_37_RF_OFFLINE` = ((1<<1) | (0<<0)), `VTSS_PORT_CLAUSE_37_RF_LINK_FAILURE` = ((0<<1) | (1<<0)), `VTSS_PORT_CLAUSE_37_RF_AUTONEG_ERROR` = ((1<<1) | (1<<0)) }
Auto-negotiation remote fault type.
- enum `vtss_port_max_tags_t` { `VTSS_PORT_MAX_TAGS_NONE`, `VTSS_PORT_MAX_TAGS_ONE`, `VTSS_PORT_MAX_TAGS_TWO` }
VLAN awareness for frame length check.
- enum `vtss_port_loop_t` { `VTSS_PORT_LOOP_DISABLE`, `VTSS_PORT_LOOP_PCS_HOST` }
Port loop back configuration.
- enum `vtss_port_forward_t` { `VTSS_PORT_FORWARD_ENABLED`, `VTSS_PORT_FORWARD_DISABLED`, `VTSS_PORT_FORWARD_INGRESS`, `VTSS_PORT_FORWARD_EGRESS` }
Port forwarding state.

Functions

- `vtss_rc vtss_port_map_set (const vtss_inst_t inst, const vtss_port_map_t port_map[VTSS_PORT_ARRAY_SIZE])`
Set port map.
- `vtss_rc vtss_port_map_get (const vtss_inst_t inst, vtss_port_map_t port_map[VTSS_PORT_ARRAY_SIZE])`
Get port map.
- `vtss_rc vtss_port_clause_37_control_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_port_clause_37_control_t *const control)`
Get clause 37 auto-negotiation Control word.
- `vtss_rc vtss_port_clause_37_control_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_port_clause_37_control_t *const control)`
Set clause 37 auto-negotiation Control word.
- `vtss_rc vtss_port_conf_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_port_conf_t *const conf)`
Set port configuration. Note: If if_type in the vtss_port_conf_t/vtss_port_interface_t definition is set to VTSS_PORT_INTERFACE_QSGMII, the ports are mapped together in groups of four. If one of the four ports is used, all four ports in the group must always be configured, but the four ports doesn't need to be configured with the same configuration. This is needed in order to achieve correct comma alignment at the QSGMII interface. Which ports that are mapped together can be found in the chip data-sheet.
- `vtss_rc vtss_port_conf_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_port_conf_t *const conf)`
Get port setup.
- `vtss_rc vtss_port_status_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_port_status_t *const status)`
Get port status.
- `vtss_rc vtss_port_counters_update (const vtss_inst_t inst, const vtss_port_no_t port_no)`
Update counters for port.
- `vtss_rc vtss_port_counters_clear (const vtss_inst_t inst, const vtss_port_no_t port_no)`
Clear counters for port.
- `vtss_rc vtss_port_counters_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_port_counters_t *const counters)`
Get counters for port.
- `vtss_rc vtss_port_basic_counters_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_basic_counters_t *const counters)`
Get basic counters for port.
- `vtss_rc vtss_port_forward_state_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_port_forward_t *const forward)`
Get port forwarding state.
- `vtss_rc vtss_port_forward_state_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_port_forward_t forward)`
Set port forwarding state.
- `vtss_rc vtss_port_ifh_conf_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_port_ifh_t *const conf)`
Set port Internal Frame Header settings.
- `vtss_rc vtss_port_ifh_conf_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_port_ifh_t *const conf)`
Get port Internal Frame Header settings.
- `vtss_rc vtss_miim_read (const vtss_inst_t inst, const vtss_chip_no_t chip_no, const vtss_miim_controller_t miim_controller, const u8 miim_addr, const u8 addr, u16 *const value)`
Direct MIIM read (bypassing port map)
- `vtss_rc vtss_miim_write (const vtss_inst_t inst, const vtss_chip_no_t chip_no, const vtss_miim_controller_t miim_controller, const u8 miim_addr, const u8 addr, const u16 value)`
Direct MIIM write (bypassing port map)

- `vtss_rc vtss_port_mmd_read` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `u8` mmd, const `u16` addr, `u16` *const value)

Read value from MMD register.
- `vtss_rc vtss_port_mmd_read_inc` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `u8` mmd, const `u16` addr, `u16` *const buf, `u8` count)

Read values (a number of 16 bit values) from MMD register.
- `vtss_rc vtss_port_mmd_write` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `u8` mmd, const `u16` addr, const `u16` value)

Write value to MMD register.
- `vtss_rc vtss_port_mmd_masked_write` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `u8` mmd, const `u16` addr, const `u16` value, const `u16` mask)

Read, modify and write value to MMD register.
- `vtss_rc vtss_mmd_read` (const `vtss_inst_t` inst, const `vtss_chip_no_t` chip_no, const `vtss_miim_controller_t` miim_controller, const `u8` miim_addr, const `u8` mmd, const `u16` addr, `u16` *const value)

Direct MMD read (Clause 45, bypassing port map)
- `vtss_rc vtss_mmd_write` (const `vtss_inst_t` inst, const `vtss_chip_no_t` chip_no, const `vtss_miim_controller_t` miim_controller, const `u8` miim_addr, const `u8` mmd, const `u16` addr, const `u16` value)

Direct MMD write (Clause 45, bypassing port map)

8.33.1 Detailed Description

Port API.

8.33.2 Macro Definition Documentation

8.33.2.1 CHIP_PORT_UNUSED

```
#define CHIP_PORT_UNUSED -1
```

Signifies an unused chip port

Definition at line 69 of file vtss_port_api.h.

8.33.2.2 VTSS_FRAME_GAP_DEFAULT

```
#define VTSS_FRAME_GAP_DEFAULT 0
```

Default frame gap used

Definition at line 203 of file vtss_port_api.h.

8.33.2.3 VTSS_MAX_FRAME_LENGTH_STANDARD

```
#define VTSS_MAX_FRAME_LENGTH_STANDARD 1518
```

IEEE 802.3 standard

Definition at line 214 of file vtss_port_api.h.

8.33.2.4 VTSS_MAX_FRAME_LENGTH_MAX [1/2]

```
#define VTSS_MAX_FRAME_LENGTH_MAX 10240
```

Maximum frame length supported for CEService

Definition at line 229 of file vtss_port_api.h.

8.33.2.5 VTSS_MAX_FRAME_LENGTH_MAX [2/2]

```
#define VTSS_MAX_FRAME_LENGTH_MAX 4776
```

Maximum frame length supported for CEService

Definition at line 229 of file vtss_port_api.h.

8.33.3 Enumeration Type Documentation

8.33.3.1 vtss_miim_controller_t

```
enum vtss_miim_controller_t
```

MII management controller.

Enumerator

VTSS_MIIM_CONTROLLER_0	MIIM controller 0
VTSS_MIIM_CONTROLLER_1	MIIM controller 1
VTSS_MIIM_CONTROLLERS	Number of MIIM controllers
VTSS_MIIM_CONTROLLER_NONE	Unassigned MIIM controller

Definition at line 42 of file vtss_port_api.h.

8.33.3.2 vtss_internal_bw_t

```
enum vtss_internal_bw_t
```

The internal bandwidth allocated for the port.

Enumerator

VTSS_BW_DEFAULT	Default to max port speed
VTSS_BW_1G	Max 1G
VTSS_BW_2500M	Max 2.5G
VTSS_BW_UNDEFINED	Undefined

Definition at line 61 of file vtss_port_api.h.

8.33.3.3 vtss_port_clause_37_remote_fault_t

```
enum vtss_port_clause_37_remote_fault_t
```

Auto-negotiation remote fault type.

Enumerator

VTSS_PORT_CLAUSE_37_RF_LINK_OK	Link OK
VTSS_PORT_CLAUSE_37_RF_OFFLINE	Off line
VTSS_PORT_CLAUSE_37_RF_LINK_FAILURE	Link failure
VTSS_PORT_CLAUSE_37_RF_AUTONEG_ERROR	Autoneg error

Definition at line 118 of file vtss_port_api.h.

8.33.3.4 vtss_port_max_tags_t

```
enum vtss_port_max_tags_t
```

VLAN awareness for frame length check.

Enumerator

VTSS_PORT_MAX_TAGS_NONE	No extra tags allowed
VTSS_PORT_MAX_TAGS_ONE	Single tag allowed
VTSS_PORT_MAX_TAGS_TWO	Single and double tag allowed

Definition at line 246 of file vtss_port_api.h.

8.33.3.5 vtss_port_loop_t

```
enum vtss_port_loop_t
```

Port loop back configuration.

Enumerator

VTSS_PORT_LOOP_DISABLE	No port loop
VTSS_PORT_LOOP_PCS_HOST	PCS host port loop

Definition at line 254 of file vtss_port_api.h.

8.33.3.6 vtss_port_forward_t

```
enum vtss_port_forward_t
```

Port forwarding state.

Enumerator

VTSS_PORT_FORWARD_ENABLED	Forward in both directions
VTSS_PORT_FORWARD_DISABLED	Forwarding and learning disabled
VTSS_PORT_FORWARD_INGRESS	Forward frames from port only
VTSS_PORT_FORWARD_EGRESS	Forward frames to port only (learning disabled)

Definition at line 398 of file vtss_port_api.h.

8.33.4 Function Documentation

8.33.4.1 vtss_port_map_set()

```
vtss_rc vtss_port_map_set (
    const vtss_inst_t inst,
    const vtss_port_map_t port_map[VTSS_PORT_ARRAY_SIZE] )
```

Set port map.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_map</i>	[IN] Port map array.

Returns

Return code.

8.33.4.2 vtss_port_map_get()

```
vtss_rc vtss_port_map_get (
    const vtss_inst_t inst,
    vtss_port_map_t port_map[VTSS_PORT_ARRAY_SIZE] )
```

Get port map.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_map</i>	[OUT] Port map.

Returns

Return code.

8.33.4.3 vtss_port_clause_37_control_get()

```
vtss_rc vtss_port_clause_37_control_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_port_clause_37_control_t *const control )
```

Get clause 37 auto-negotiation Control word.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>control</i>	[OUT] Control structure.

Returns

Return code.

8.33.4.4 vtss_port_clause_37_control_set()

```
vtss_rc vtss_port_clause_37_control_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_port_clause_37_control_t *const control )
```

Set clause 37 auto-negotiation Control word.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>control</i>	[IN] Control structure.

Returns

Return code.

8.33.4.5 vtss_port_conf_set()

```
vtss_rc vtss_port_conf_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_port_conf_t *const conf )
```

Set port configuration. Note: If if_type in the vtss_port_conf_t/vtss_port_interface_t definition is set to VTSS_<→PORT_INTERFACE_QSGMII, the ports are mapped together in groups of four. If one of the four ports is used, all four ports in the group must always be configured, but the four ports doesn't need to be configured with the same configuration. This is needed in order to achieve correct comma alignment at the QSGMII interface. Which ports that are mapped together can be found in the chip data-sheet.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>conf</i>	[IN] Port setup structure.

Returns

Return code.

8.33.4.6 `vtss_port_conf_get()`

```
vtss_rc vtss_port_conf_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_port_conf_t *const conf )
```

Get port setup.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>conf</i>	[OUT] Port configuration.

Returns

Return code.

8.33.4.7 `vtss_port_status_get()`

```
vtss_rc vtss_port_status_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_port_status_t *const status )
```

Get port status.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>status</i>	[OUT] Status structure.

Returns

Return code.

8.33.4.8 `vtss_port_counters_update()`

```
vtss_rc vtss_port_counters_update (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no )
```

Update counters for port.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.

Returns

Return code.

8.33.4.9 vtss_port_counters_clear()

```
vtss_rc vtss_port_counters_clear (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no )
```

Clear counters for port.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port/aggregation number.

Returns

Return code.

8.33.4.10 vtss_port_counters_get()

```
vtss_rc vtss_port_counters_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_port_counters_t *const counters )
```

Get counters for port.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port/aggregation number.
<i>counters</i>	[OUT] Counter structure.

Returns

Return code.

8.33.4.11 vtss_port_basic_counters_get()

```
vtss_rc vtss_port_basic_counters_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_basic_counters_t *const counters )
```

Get basic counters for port.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port/aggregation number.
<i>counters</i>	[OUT] Counter structure.

Returns

Return code.

8.33.4.12 vtss_port_forward_state_get()

```
vtss_rc vtss_port_forward_state_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_port_forward_t *const forward )
```

Get port forwarding state.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>forward</i>	[OUT] Forwarding state.

Returns

Return code.

8.33.4.13 vtss_port_forward_state_set()

```
vtss_rc vtss_port_forward_state_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_port_forward_t forward )
```

Set port forwarding state.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>forward</i>	[IN] Forwarding state.

Returns

Return code.

8.33.4.14 vtss_port_ifh_conf_set()

```
vtss_rc vtss_port_ifh_conf_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_port_ifh_t *const conf )
```

Set port Internal Frame Header settings.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>conf</i>	[IN] Port IFH structure.

Returns

Return code.

8.33.4.15 vtss_port_ifh_conf_get()

```
vtss_rc vtss_port_ifh_conf_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_port_ifh_t *const conf )
```

Get port Internal Frame Header settings.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>conf</i>	[OUT] Port IFH configuration.

Returns

Return code.

8.33.4.16 vtss_miim_read()

```
vtss_rc vtss_miim_read (
    const vtss_inst_t inst,
    const vtss_chip_no_t chip_no,
    const vtss_miim_controller_t miim_controller,
    const u8 miim_addr,
    const u8 addr,
    u16 *const value )
```

Direct MIIM read (bypassing port map)

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>chip_no</i>	[IN] Chip number (if multi-chip instance).
<i>miim_controller</i>	[IN] MIIM Controller Instance
<i>miim_addr</i>	[IN] MIIM Device Address
<i>addr</i>	[IN] MIIM Register Address
<i>value</i>	[OUT] Register value read

Returns

Return code.

8.33.4.17 vtss_miim_write()

```
vtss_rc vtss_miim_write (
    const vtss_inst_t inst,
    const vtss_chip_no_t chip_no,
    const vtss_miim_controller_t miim_controller,
    const u8 miim_addr,
    const u8 addr,
    const u16 value )
```

Direct MIIM write (bypassing port map)

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>chip_no</i>	[IN] Chip number (if multi-chip instance).
<i>miim_controller</i>	[IN] MIIM Controller Instance
<i>miim_addr</i>	[IN] MIIM Device Address
<i>addr</i>	[IN] MIIM Register Address
<i>value</i>	[IN] Register value to write

Returns

Return code.

8.33.4.18 vtss_port_mmd_read()

```
vtss_rc vtss_port_mmd_read (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const u8 mmd,
    const u16 addr,
    u16 *const value )
```

Read value from MMD register.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number connected to MMD.
<i>mmd</i>	[IN] MMD number.
<i>addr</i>	[IN] PHY register address.
<i>value</i>	[OUT] PHY register value.

Returns

Return code.

8.33.4.19 vtss_port_mmd_read_inc()

```
vtss_rc vtss_port_mmd_read_inc (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const u8 mmd,
    const u16 addr,
    u16 *const buf,
    u8 count )
```

Read values (a number of 16 bit values) from MMD register.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number connected to MMD.
<i>mmd</i>	[IN] MMD number.
<i>addr</i>	[IN] PHY register address.
<i>buf</i>	[OUT] PHY register values.
<i>count</i>	[IN] number of values to read.

Returns

Return code.

8.33.4.20 vtss_port_mmd_write()

```
vtss_rc vtss_port_mmd_write (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const u8 mmd,
    const u16 addr,
    const u16 value )
```

Write value to MMD register.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number connected to MMD.
<i>mmd</i>	[IN] MMD number.
<i>addr</i>	[IN] PHY register address.
<i>value</i>	[IN] PHY register value.

Returns

Return code.

8.33.4.21 vtss_port_mmd_masked_write()

```
vtss_rc vtss_port_mmd_masked_write (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const u8 mmd,
    const u16 addr,
    const u16 value,
    const u16 mask )
```

Read, modify and write value to MMD register.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number connected to MMD.
<i>mmd</i>	[IN] MMD number.
<i>addr</i>	[IN] PHY register address.
<i>value</i>	[IN] PHY register value.
<i>mask</i>	[IN] PHY register mask, only enabled bits are changed.

Returns

Return code.

8.33.4.22 vtss_mmd_read()

```
vtss_rc vtss_mmd_read (
    const vtss_inst_t inst,
    const vtss_chip_no_t chip_no,
    const vtss_miim_controller_t miim_controller,
    const u8 miim_addr,
    const u8 mmd,
    const u16 addr,
    u16 *const value )
```

Direct MMD read (Clause 45, bypassing port map)

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>chip_no</i>	[IN] Chip number (if multi-chip instance).
<i>miim_controller</i>	[IN] MIIM Controller Instance
<i>miim_addr</i>	[IN] MIIM Device Address
<i>mmd</i>	[IN] MMD number.
<i>addr</i>	[IN] MIIM Register Address
<i>value</i>	[OUT] Register value read

Returns

Return code.

8.33.4.23 vtss_mmd_write()

```
vtss_rc vtss_mmd_write (
    const vtss_inst_t inst,
```

```

const vtss_chip_no_t chip_no,
const vtss_miim_controller_t miim_controller,
const u8 miim_addr,
const u8 mmd,
const u16 addr,
const u16 value )

```

Direct MMD write (Clause 45, bypassing port map)

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>chip_no</i>	[IN] Chip number (if multi-chip instance).
<i>miim_controller</i>	[IN] MIIM Controller Instance
<i>miim_addr</i>	[IN] MIIM Device Address
<i>mmd</i>	[IN] MMD number.
<i>addr</i>	[IN] MIIM Register Address
<i>value</i>	[IN] Register value to write

Returns

Return code.

8.34 vtss_api/include/vtss_qos_api.h File Reference

QoS API.

```
#include <vtss/api/types.h>
```

Data Structures

- struct [vtss_red_t](#)
Random Early Detection configuration struct version 1 (per port, per queue)
- struct [vtss_qos_conf_t](#)
All parameters below are defined per chip.
- struct [vtss_policer_t](#)
Policer.
- struct [vtss_policer_ext_t](#)
Policer Extensions.
- struct [vtss_dlb_policer_conf_t](#)
Dual leaky buckets policer configuration.
- struct [vtss_shaper_t](#)
Shaper.
- struct [vtss_qos_port_conf_t](#)
QoS setup per port.
- struct [vtss_qce_mac_t](#)
QCE MAC information.
- struct [vtss_qce_tag_t](#)

- *QCE tag information.*
- struct `vtss_qce_frame_etype_t`
Frame data for VTSS_QCE_TYPEETYPE.
- struct `vtss_qce_frame_llc_t`
Frame data for VTSS_QCE_TYPELLC.
- struct `vtss_qce_frame_snap_t`
Frame data for VTSS_QCE_TYPESNAP.
- struct `vtss_qce_frame_ipv4_t`
Frame data for VTSS_QCE_TYPEIPV4.
- struct `vtss_qce_frame_ipv6_t`
Frame data for VTSS_QCE_TYPEIPV6.
- struct `vtss_qce_key_t`
QCE key.
- struct `vtss_qce_action_t`
QCE action.
- struct `vtss_qce_t`
QoS Control Entry.

Macros

- `#define VTSS_PORT_POLICERS 4`
- `#define VTSS_PORT_POLICER_CPU_QUEUES 8`
- `#define VTSS_QCL_IDS 1`
- `#define VTSS_QCL_ID_START 0`
- `#define VTSS_QCL_ID_END (VTSS_QCL_ID_START + VTSS_QCL_IDS)`
- `#define VTSS_QCL_ARRAY_SIZE VTSS_QCL_ID_END`
- `#define VTSS_QCE_ID_LAST 0`

Typedefs

- `typedef u32 vtss_qcl_id_t`
QCL ID type.

Enumerations

- enum `vtss_tag_remark_mode_t`{ `VTSS_TAG_REMARK_MODE_CLASSIFIED` = 0, `VTSS_TAG_REMARK_MODE_DEFAULT` = 2, `VTSS_TAG_REMARK_MODE_MAPPED` = 3 }
Tag Remark Mode.
- enum `vtss_dscp_mode_t`{ `VTSS_DSCP_MODE_NONE`, `VTSS_DSCP_MODE_ZERO`, `VTSS_DSCP_MODE_SEL`, `VTSS_DSCP_MODE_ALL` }
DSCP mode for ingress port.
- enum `vtss_dscpemode_t` { `VTSS_DSCP_EMODE_DISABLE`, `VTSS_DSCP_EMODE_REMARK`, `VTSS_DSCP_EMODE_REMAP` }
DSCP mode for egress port.
- enum `vtss_qce_type_t`{
 `VTSS_QCE_TYPE_ANY`, `VTSS_QCE_TYPEETYPE`, `VTSS_QCE_TYPE_LLCC`, `VTSS_QCE_TYPE_SNAP`,
 `VTSS_QCE_TYPE_IPV4`, `VTSS_QCE_TYPE_IPV6` }
QoS Control Entry type.

Functions

- `vtss_rc vtss_qos_conf_get` (const `vtss_inst_t` inst, `vtss_qos_conf_t` *const conf)
Get QoS setup for switch.
- `vtss_rc vtss_qos_conf_set` (const `vtss_inst_t` inst, const `vtss_qos_conf_t` *const conf)
Set QoS setup for switch.
- `vtss_rc vtss_qos_port_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_qos_port_conf_t` *const conf)
Get QoS setup for port.
- `vtss_rc vtss_qos_port_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_qos_port_conf_t` *const conf)
Set QoS setup for port.
- `vtss_rc vtss_qce_init` (const `vtss_inst_t` inst, const `vtss_qce_type_t` type, `vtss_qce_t` *const qce)
Initialize QCE to default values.
- `vtss_rc vtss_qce_add` (const `vtss_inst_t` inst, const `vtss_qcl_id_t` qcl_id, const `vtss_qce_id_t` qce_id, const `vtss_qce_t` *const qce)
Add QCE to QCL.
- `vtss_rc vtss_qce_del` (const `vtss_inst_t` inst, const `vtss_qcl_id_t` qcl_id, const `vtss_qce_id_t` qce_id)
Delete QCE from QCL.

8.34.1 Detailed Description

QoS API.

This header file describes Quality of Service functions

8.34.2 Macro Definition Documentation

8.34.2.1 VTSS_PORT_POLICERS

```
#define VTSS_PORT_POLICERS 4
```

Number of port policers

Definition at line 177 of file vtss_qos_api.h.

8.34.2.2 VTSS_PORT_POLICER_CPU_QUEUES

```
#define VTSS_PORT_POLICER_CPU_QUEUES 8
```

Number of cpu queues pr port policer

Definition at line 193 of file vtss_qos_api.h.

8.34.2.3 VTSS_QCL_IDS

```
#define VTSS_QCL_IDS 1
```

Number of QCLs

Definition at line 452 of file vtss_qos_api.h.

8.34.2.4 VTSS_QCL_ID_START

```
#define VTSS_QCL_ID_START 0
```

QCL ID start number

Definition at line 453 of file vtss_qos_api.h.

8.34.2.5 VTSS_QCL_ID_END

```
#define VTSS_QCL_ID_END (VTSS_QCL_ID_START + VTSS_QCL_IDS)
```

QCL ID end number

Definition at line 454 of file vtss_qos_api.h.

8.34.2.6 VTSS_QCL_ARRAY_SIZE

```
#define VTSS_QCL_ARRAY_SIZE VTSS_QCL_ID_END
```

QCL ID array size

Definition at line 455 of file vtss_qos_api.h.

8.34.2.7 VTSS_QCE_ID_LAST

```
#define VTSS_QCE_ID_LAST 0
```

Special value used to add last in list

Definition at line 457 of file vtss_qos_api.h.

8.34.3 Enumeration Type Documentation

8.34.3.1 vtss_tag_remark_mode_t

```
enum vtss_tag_remark_mode_t
```

Tag Remark Mode.

Enumerator

VTSS_TAG_REMARK_MODE_CLASSIFIED	Use classified PCP/DEI values
VTSS_TAG_REMARK_MODE_DEFAULT	Use default (configured) PCP/DEI values
VTSS_TAG_REMARK_MODE_MAPPED	Use mapped versions of classified QOS class and DP level

Definition at line 312 of file vtss_qos_api.h.

8.34.3.2 vtss_dscp_mode_t

```
enum vtss\_dscp\_mode\_t
```

DSCP mode for ingress port.

Enumerator

VTSS_DSCP_MODE_NONE	DSCP not remarked
VTSS_DSCP_MODE_ZERO	DSCP value zero remarked
VTSS_DSCP_MODE_SEL	DSCP values selected above (dscp_remark) are remarked
VTSS_DSCP_MODE_ALL	DSCP remarked for all values

Definition at line 322 of file vtss_qos_api.h.

8.34.3.3 vtss_dscp_emode_t

```
enum vtss\_dscp\_emode\_t
```

DSCP mode for egress port.

Enumerator

VTSS_DSCP_EMODE_DISABLE	DSCP not remarked
VTSS_DSCP_EMODE_REMARK	DSCP remarked with DSCP value from analyzer
VTSS_DSCP_EMODE_REMAP	DSCP remarked with DSCP value from analyzer remapped through global remap table

Definition at line 333 of file vtss_qos_api.h.

8.34.3.4 vtss_qce_type_t

```
enum vtss\_qce\_type\_t
```

QoS Control Entry type.

Enumerator

VTSS_QCE_TYPE_ANY	Any frame type
VTSS_QCE_TYPE_ETYPE	Ethernet Type
VTSS_QCE_TYPE_LLC	LLC
VTSS_QCE_TYPE_SNAP	SNAP
VTSS_QCE_TYPE_IPV4	IPv4
VTSS_QCE_TYPE_IPV6	IPv6

Definition at line 460 of file vtss_qos_api.h.

8.34.4 Function Documentation

8.34.4.1 vtss_qos_conf_get()

```
vtss_rc vtss_qos_conf_get (
    const vtss_inst_t inst,
    vtss_qos_conf_t *const conf )
```

Get QoS setup for switch.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>conf</i>	[OUT] QoS setup structure.

Returns

Return code.

8.34.4.2 vtss_qos_conf_set()

```
vtss_rc vtss_qos_conf_set (
    const vtss_inst_t inst,
    const vtss_qos_conf_t *const conf )
```

Set QoS setup for switch.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>conf</i>	[IN] QoS setup structure.

Returns

Return code.

8.34.4.3 vtss_qos_port_conf_get()

```
vtss_rc vtss_qos_port_conf_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_qos_port_conf_t *const conf )
```

Get QoS setup for port.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>conf</i>	[OUT] QoS setup structure.

Returns

Return code.

8.34.4.4 vtss_qos_port_conf_set()

```
vtss_rc vtss_qos_port_conf_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_qos_port_conf_t *const conf )
```

Set QoS setup for port.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>conf</i>	[IN] QoS setup structure.

Returns

Return code.

8.34.4.5 vtss_qce_init()

```
vtss_rc vtss_qce_init (
    const vtss_inst_t inst,
    const vtss_qce_type_t type,
    vtss_qce_t *const qce )
```

Initialize QCE to default values.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>type</i>	[IN] QCE type.
<i>qce</i>	[OUT] QCE structure.

Returns

Return code.

8.34.4.6 vtss_qce_add()

```
vtss_rc vtss_qce_add (
    const vtss_inst_t inst,
    const vtss_qcl_id_t qcl_id,
    const vtss_qce_id_t qce_id,
    const vtss_qce_t *const qce )
```

Add QCE to QCL.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>qcl_id</i>	[IN] QCL ID.
<i>qce_id</i>	[IN] QCE ID. The QCE will be added before the entry with this ID. VTSS_QCE_ID_LAST is reserved for inserting last.
<i>qce</i>	[IN] QCE structure.

Returns

Return code.

8.34.4.7 vtss_qce_del()

```
vtss_rc vtss_qce_del (
    const vtss_inst_t inst,
```

```
const vtss_qcl_id_t qcl_id,  
const vtss_qce_id_t qce_id )
```

Delete QCE from QCL.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>qcl_id</i>	[IN] QCL ID.
<i>qce_id</i>	[IN] QCE ID.

Returns

Return code.

8.35 vtss_api/include/vtss_rab_api.h File Reference

RAB API.

```
#include <vtss/api/types.h>
```

8.35.1 Detailed Description

RAB API.

8.36 vtss_api/include/vtss_security_api.h File Reference

Security API.

```
#include <vtss/api/types.h>
```

Data Structures

- struct [vtss_acl_policer_conf_t](#)
ACL policer configuration.
- struct [vtss_acl_action_t](#)
ACL Action.
- struct [vtss_acl_port_conf_t](#)
ACL port configuration.
- struct [vtss_ace_vlan_t](#)
ACE VLAN information.
- struct [vtss_ace_frame_etype_t](#)
Frame data for VTSS_ACE_TYPE_ETYPE.
- struct [vtss_ace_frame_llc_t](#)
Frame data for VTSS_ACE_TYPE_LLCC.
- struct [vtss_ace_frame_snap_t](#)
Frame data for VTSS_ACE_TYPE_SNAP.
- struct [vtss_ace_frame_arp_t](#)
Frame data for VTSS_ACE_TYPE_ARP.
- struct [vtss_ace_frame_ipv4_t](#)
Frame data for VTSS_ACE_TYPE_IPV4.
- struct [vtss_ace_frame_ipv6_t](#)
Frame data for VTSS_ACE_TYPE_IPV6.
- struct [vtss_ace_t](#)
Access Control Entry.

Macros

- `#define VTSS_ACE_ID_LAST 0`
- `#define VTSS_PORT_NO_ANY VTSS_PORT_NO_NONE`

Typedefs

- `typedef u32 vtss_acl_port_counter_t`
ACL port counter.
- `typedef u32 vtss_ace_id_t`
ACE ID type.
- `typedef vtss_vcap_u8_t vtss_ace_u8_t`
ACE 8 bit value and mask.
- `typedef vtss_vcap_u16_t vtss_ace_u16_t`
ACE 16 bit value and mask.
- `typedef vtss_vcap_u32_t vtss_ace_u32_t`
ACE 32 bit value and mask.
- `typedef vtss_vcap_u40_t vtss_ace_u40_t`
ACE 40 bit value and mask.
- `typedef vtss_vcap_u48_t vtss_ace_u48_t`
ACE 48 bit value and mask.
- `typedef vtss_vcap_u128_t vtss_ace_u128_t`
ACE 128 bit value and mask.
- `typedef vtss_vcap_vid_t vtss_ace_vid_t`
ACE VLAN ID value and mask.
- `typedef vtss_vcap_ip_t vtss_ace_ip_t`
ACE IP address value and mask.
- `typedef vtss_vcap_udp_tcp_t vtss_ace_udp_tcp_t`
ACE UDP/TCP port range.
- `typedef u32 vtss_ace_counter_t`
ACE hit counter.

Enumerations

- enum `vtss_auth_state_t`{ `VTSS_AUTH_STATE_NONE`, `VTSS_AUTH_STATE_EGRESS`, `VTSS_AUTH_STATE_BOTH` }
- Authentication state.*
- enum `vtss_ace_type_t`{
 `VTSS_ACE_TYPE_ANY`, `VTSS_ACE_TYPEETYPE`, `VTSS_ACE_TYPE_LLIC`, `VTSS_ACE_TYPE_SNAP`,
 `VTSS_ACE_TYPE_ARP`, `VTSS_ACE_TYPE_IPV4`, `VTSS_ACE_TYPE_IPV6` }
- ACE frame type.*
- enum `vtss_ace_bit_t`{ `VTSS_ACE_BIT_ANY`, `VTSS_ACE_BIT_0`, `VTSS_ACE_BIT_1` }
- ACE 1 bit.*

Functions

- `vtss_rc vtss_auth_port_state_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_auth_state_t` *const state)
Get 802.1X Authentication state for a port.
- `vtss_rc vtss_auth_port_state_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_auth_state_t` state)
Set 802.1X Authentication state for a port.
- `vtss_rc vtss_acl_policer_conf_get` (const `vtss_inst_t` inst, const `vtss_acl_policer_no_t` policer_no, `vtss_acl_policer_conf_t` *const conf)
Get ACL policer configuration.
- `vtss_rc vtss_acl_policer_conf_set` (const `vtss_inst_t` inst, const `vtss_acl_policer_no_t` policer_no, const `vtss_acl_policer_conf_t` *const conf)
Set ACL policer configuration.
- `vtss_rc vtss_acl_port_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_acl_port_conf_t` *const conf)
Get ACL configuration for port.
- `vtss_rc vtss_acl_port_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_acl_port_conf_t` *const conf)
Set ACL configuration for port.
- `vtss_rc vtss_acl_port_counter_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_acl_port_counter_t` *const counter)
Get default action counter for port.
- `vtss_rc vtss_acl_port_counter_clear` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no)
Clear default action counter for port.
- `vtss_rc vtss_ace_init` (const `vtss_inst_t` inst, const `vtss_ace_type_t` type, `vtss_ace_t` *const ace)
Initialize ACE to default values.
- `vtss_rc vtss_ace_add` (const `vtss_inst_t` inst, const `vtss_ace_id_t` ace_id_next, const `vtss_ace_t` *const ace)
Add/modify ACE.
- `vtss_rc vtss_ace_del` (const `vtss_inst_t` inst, const `vtss_ace_id_t` ace_id)
Delete ACE.
- `vtss_rc vtss_ace_counter_get` (const `vtss_inst_t` inst, const `vtss_ace_id_t` ace_id, `vtss_ace_counter_t` *const counter)
Get ACE counter.
- `vtss_rc vtss_ace_counter_clear` (const `vtss_inst_t` inst, const `vtss_ace_id_t` ace_id)
Clear ACE counter.

8.36.1 Detailed Description

Security API.

This header file describes security functions

8.36.2 Macro Definition Documentation

8.36.2.1 VTSS_ACE_ID_LAST

```
#define VTSS_ACE_ID_LAST 0
```

Special value used to add last in list

Definition at line 352 of file vtss_security_api.h.

8.36.2.2 VTSS_PORT_NO_ANY

```
#define VTSS_PORT_NO_ANY VTSS_PORT_NO_NONE
```

Any port number

Definition at line 407 of file vtss_security_api.h.

8.36.3 Enumeration Type Documentation

8.36.3.1 vtss_auth_state_t

```
enum vtss_auth_state_t
```

Authentication state.

Enumerator

VTSS_AUTH_STATE_NONE	Not authenticated
VTSS_AUTH_STATE_EGRESS	Authenticated in egress direction
VTSS_AUTH_STATE_BOTH	Authenticated in both directions

Definition at line 59 of file vtss_security_api.h.

8.36.3.2 vtss_ace_type_t

```
enum vtss_ace_type_t
```

ACE frame type.

Enumerator

VTSS_ACE_TYPE_ANY	Any frame type
VTSS_ACE_TYPE_ETYPE	Ethernet Type

Enumerator

VTSS_ACE_TYPE_LLC	LLC
VTSS_ACE_TYPE_SNAP	SNAP
VTSS_ACE_TYPE_ARP	ARP/RARP
VTSS_ACE_TYPE_IPV4	IPv4
VTSS_ACE_TYPE_IPV6	IPv6

Definition at line 338 of file vtss_security_api.h.

8.36.3.3 vtss_ace_bit_t

enum [vtss_ace_bit_t](#)

ACE 1 bit.

Enumerator

VTSS_ACE_BIT_ANY	Value 0 or 1
VTSS_ACE_BIT_0	Value 0
VTSS_ACE_BIT_1	Value 1

Definition at line 355 of file vtss_security_api.h.

8.36.4 Function Documentation

8.36.4.1 vtss_auth_port_state_get()

```
vtss_rc vtss_auth_port_state_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_auth_state_t *const state )
```

Get 802.1X Authentication state for a port.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>state</i>	[OUT] Authentication state.

Returns

Return code.

8.36.4.2 vtss_auth_port_state_set()

```
vtss_rc vtss_auth_port_state_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_auth_state_t state )
```

Set 802.1X Authentication state for a port.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>state</i>	[IN] Authentication state.

Returns

Return code.

8.36.4.3 vtss_acl_policer_conf_get()

```
vtss_rc vtss_acl_policer_conf_get (
    const vtss_inst_t inst,
    const vtss_acl_policer_no_t policer_no,
    vtss_acl_policer_conf_t *const conf )
```

Get ACL policer configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>policer_no</i>	[IN] ACL policer number.
<i>conf</i>	[OUT] ACL policer configuration.

Returns

Return code.

8.36.4.4 vtss_acl_policer_conf_set()

```
vtss_rc vtss_acl_policer_conf_set (
    const vtss_inst_t inst,
    const vtss_acl_policer_no_t policer_no,
    const vtss_acl_policer_conf_t *const conf )
```

Set ACL policer configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>policer_no</i>	[IN] ACL policer number.
<i>conf</i>	[IN] ACL policer configuration.

Returns

Return code.

8.36.4.5 vtss_acl_port_conf_get()

```
vtss_rc vtss_acl_port_conf_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_acl_port_conf_t *const conf )
```

Get ACL configuration for port.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>conf</i>	[OUT] Port configuration.

Returns

Return code.

8.36.4.6 vtss_acl_port_conf_set()

```
vtss_rc vtss_acl_port_conf_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_acl_port_conf_t *const conf )
```

Set ACL configuration for port.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>conf</i>	[IN] Port configuration.

Returns

Return code.

8.36.4.7 vtss_acl_port_counter_get()

```
vtss_rc vtss_acl_port_counter_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_acl_port_counter_t *const counter )
```

Get default action counter for port.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>counter</i>	[OUT] Default action counter for port.

Returns

Return code.

8.36.4.8 vtss_acl_port_counter_clear()

```
vtss_rc vtss_acl_port_counter_clear (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no )
```

Clear default action counter for port.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.

Returns

Return code.

8.36.4.9 vtss_ace_init()

```
vtss_rc vtss_ace_init (
    const vtss_inst_t inst,
    const vtss_ace_type_t type,
    vtss_ace_t *const ace )
```

Initialize ACE to default values.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>type</i>	[IN] ACE type.
<i>ace</i>	[OUT] ACE structure.

Returns

Return code.

8.36.4.10 vtss_ace_add()

```
vtss_rc vtss_ace_add (
    const vtss_inst_t inst,
    const vtss_ace_id_t ace_id_next,
    const vtss_ace_t *const ace )
```

Add/modify ACE.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>ace_id_next</i>	[IN] ACE ID of next entry. The ACE will be added before the entry with this ID. VTSS_ACE_ID_LAST is reserved for inserting last.
<i>ace</i>	[IN] ACE structure.

Returns

Return code.

8.36.4.11 vtss_ace_del()

```
vtss_rc vtss_ace_del (
    const vtss_inst_t inst,
    const vtss_ace_id_t ace_id )
```

Delete ACE.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>ace_id</i>	[IN] ACE ID.

Returns

Return code.

8.36.4.12 vtss_ace_counter_get()

```
vtss_rc vtss_ace_counter_get (
    const vtss_inst_t inst,
    const vtss_ace_id_t ace_id,
    vtss_ace_counter_t *const counter )
```

Get ACE counter.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>ace_id</i>	[IN] ACE ID.
<i>counter</i>	[OUT] ACE counter.

Returns

Return code.

8.36.4.13 vtss_ace_counter_clear()

```
vtss_rc vtss_ace_counter_clear (
    const vtss_inst_t inst,
    const vtss_ace_id_t ace_id )
```

Clear ACE counter.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>ace_id</i>	[IN] ACE ID.

Returns

Return code.

8.37 vtss_api/include/vtss_sfi4_api.h File Reference

SFI4 API.

```
#include <vtss/api/types.h>
```

8.37.1 Detailed Description

SFI4 API.

8.38 vtss_api/include/vtss_sync_api.h File Reference

Synchronization API.

```
#include "vtss/api/types.h"
```

Data Structures

- struct [vtss_sync_clock_out_t](#)
Struct containing configuration for a recovered clock output port.
- struct [vtss_sync_clock_in_t](#)
Struct containing configuration selecting the recovered input clock port, to be delivered to a selected output clock port.

Macros

- #define [VTSS_SYNC_CLK_A](#) 0
- #define [VTSS_SYNC_CLK_B](#) 1
- #define [VTSS_SYNC_CLK_MAX](#) 2

Typedefs

- typedef [u32 vtss_sync_clk_port_t](#)
Identification of a output clock port.

Enumerations

- enum `vtss_sync_clock_out_t`{ `VTSS_SYNCE_DIVIDER_1`, `VTSS_SYNCE_DIVIDER_4`, `VTSS_SYNCE_DIVIDER_5` }

Identification of a Clock dividing value used when selected input clock goes to output.

Functions

- `vtss_rc vtss_sync_clock_out_set` (const `vtss_inst_t` inst, const `vtss_sync_clk_port_t` clk_port, const `vtss_sync_clock_out_t` *const conf)
Set the configuration of a selected output clock port - against external clock controller.
- `vtss_rc vtss_sync_clock_out_get` (const `vtss_inst_t` inst, const `vtss_sync_clk_port_t` clk_port, `vtss_sync_clock_out_t` *const conf)
Get the configuration of a selected output clock port - against external clock controller.
- `vtss_rc vtss_sync_clock_in_set` (const `vtss_inst_t` inst, const `vtss_sync_clk_port_t` clk_port, const `vtss_sync_clock_in_t` *const conf)
Set the configuration of input port for a selected output clock port.
- `vtss_rc vtss_sync_clock_in_get` (const `vtss_inst_t` inst, const `vtss_sync_clk_port_t` clk_port, `vtss_sync_clock_in_t` *const conf)
Get the configuration of input port for a selected output clock port.

8.38.1 Detailed Description

Synchronization API.

This header file describes synchronization functions

8.38.2 Macro Definition Documentation

8.38.2.1 VTSS_SYNCE_CLK_A

```
#define VTSS_SYNCE_CLK_A 0
```

Clock A output port

Definition at line 56 of file vtss_sync_api.h.

8.38.2.2 VTSS_SYNCE_CLK_B

```
#define VTSS_SYNCE_CLK_B 1
```

Clock B output port

Definition at line 57 of file vtss_sync_api.h.

8.38.2.3 VTSS_SYNCE_CLK_MAX

```
#define VTSS_SYNCE_CLK_MAX 2
```

Number of recovered clock outputs

Definition at line 61 of file vtss_sync_api.h.

8.38.3 Enumeration Type Documentation

8.38.3.1 vtss_sync_divider_t

```
enum vtss_sync_divider_t
```

Identification of a Clock dividing value used when selected input clock goes to output.

Enumerator

VTSS_SYNCE_DIVIDER_1	Divide input clock with one (no division)
VTSS_SYNCE_DIVIDER_4	Divide input clock with 4
VTSS_SYNCE_DIVIDER_5	Divide input clock with 5

Definition at line 65 of file vtss_sync_api.h.

8.38.4 Function Documentation

8.38.4.1 vtss_sync_clock_out_set()

```
vtss_rc vtss_sync_clock_out_set (
    const vtss_inst_t inst,
    const vtss_sync_clk_port_t clk_port,
    const vtss_sync_clock_out_t *const conf )
```

Set the configuration of a selected output clock port - against external clock controller.

Parameters

<i>inst</i>	[IN] handle to an API instance.
<i>clk_port</i>	[IN] Selection of the output clock port (VTSS_SYNCE_CLK_A or VTSS_SYNCE_CLK_B)
<i>conf</i>	[IN] pointer to a vtss_sync_clock_out_t configuration structure.

Returns

Return code.

8.38.4.2 vtss_sync_clock_out_get()

```
vtss_rc vtss_sync_clock_out_get (
    const vtss_inst_t inst,
    const vtss_sync_clk_port_t clk_port,
    vtss_sync_clock_out_t *const conf )
```

Get the configuration of a selected output clock port - against external clock controller.

Parameters

<i>inst</i>	[IN] handle to an API instance.
<i>clk_port</i>	[IN] Selection of the output clock port (VTSS_SYNC_CLK_A or VTSS_SYNC_CLK_B)
<i>conf</i>	[IN] pointer to a vtss_sync_clock_out_t configuration structure.

Returns

Return code.

8.38.4.3 vtss_sync_clock_in_set()

```
vtss_rc vtss_sync_clock_in_set (
    const vtss_inst_t inst,
    const vtss_sync_clk_port_t clk_port,
    const vtss_sync_clock_in_t *const conf )
```

Set the configuration of input port for a selected output clock port.

Parameters

<i>inst</i>	[IN] handle to an API instance.
<i>clk_port</i>	[IN] Selection of the output clock port (VTSS_SYNC_CLK_A or VTSS_SYNC_CLK_B)
<i>conf</i>	[IN] pointer to a vtss_sync_clock_in_t configuration structure.

Returns

Return code.

8.38.4.4 vtss_sync_clock_in_get()

```
vtss_rc vtss_sync_clock_in_get (
    const vtss_inst_t inst,
    const vtss_sync_clk_port_t clk_port,
    vtss_sync_clock_in_t *const conf )
```

Get the configuration of input port for a selected output clock port.

Parameters

<i>inst</i>	[IN] handle to an API instance.
<i>clk_port</i>	[IN] Selection of the output clock port (VTSS_SYNC_CLK_A or VTSS_SYNC_CLK_B)
<i>conf</i>	[IN] pointer to a vtss_sync_clock_in_t configuration structure.

Returns

Return code.

8.39 vtss_api/include/vtss_tfi5_api.h File Reference

TFI5 API.

```
#include <vtss/api/types.h>
```

8.39.1 Detailed Description

TFI5 API.

8.40 vtss_api/include/vtss_ts_api.h File Reference

TimeStamping API.

```
#include <vtss/api/types.h>
```

Data Structures

- struct [vtss_ts_ext_clock_mode_t](#)
external clock output configuration.
- struct [vtss_ts_operation_mode_t](#)
Timestamp operation.
- struct [vtss_ts_internal_mode_t](#)
Hardware timestamping format mode for internal ports.
- struct [vtss_ts_id_t](#)
Timestamp identifier.
- struct [vtss_ts_timestamp_t](#)
Timestamp structure.
- struct [vtss_ts_timestamp_alloc_t](#)
Timestamp allocation.

Macros

- `#define VTSS_HW_TIME_CNT_PR_SEC 1000000000`
Number of clock cycle counts pr sec.
- `#define VTSS_HW_TIME_NSEC_PR_CNT 1`
Number of nanoseconds pr clock count.
- `#define VTSS_HW_TIME_WRAP_LIMIT VTSS_HW_TIME_CNT_PR_SEC /* time counter wrap around limit+1 */`
Jaguar nanosecond time counter wrap around value (jaguar time counter wraps each second).
- `#define VTSS_HW_TIME_MIN_ADJ_RATE 40 /* 4 ppb */`
Jaguar/Luton26 minimum adjustment rate in units of 0,1 ppb.
- `#define VTSS_HW_TIME_MAX_FINE_ADJ 25`
This is the max time offset adjustment that os done without setting ports in disabled state.

Typedefs

- `typedef struct vtss_ts_ext_clock_mode_t vtss_ts_ext_clock_mode_t`
external clock output configuration.
- `typedef struct vtss_ts_operation_mode_t vtss_ts_operation_mode_t`
Timestamp operation.
- `typedef struct vtss_ts_internal_mode_t vtss_ts_internal_mode_t`
Hardware timestamping format mode for internal ports.
- `typedef struct vtss_ts_id_t vtss_ts_id_t`
Timestamp identifier.
- `typedef struct vtss_ts_timestamp_t vtss_ts_timestamp_t`
Timestamp structure.
- `typedef struct vtss_ts_timestamp_alloc_t vtss_ts_timestamp_alloc_t`
Timestamp allocation.

Enumerations

- `enum vtss_ts_ext_clock_one_pps_mode_t { TS_EXT_CLOCK_MODE_ONE_PPS_DISABLE, TS_EXT_CLOCK_MODE_ONE_PPS_OUTPUT, TS_EXT_CLOCK_MODE_ONE_PPS_INPUT, TS_EXT_CLOCK_MODE_ONE_PPS_OUTPUT_INPUT, TS_EXT_CLOCK_MODE_MAX }`
parameter for setting the external clock mode.
- `enum vtss_ts_mode_t { TS_MODE_NONE, TS_MODE_EXTERNAL, TS_MODE_INTERNAL, TX_MODE_MAX }`
parameter for setting the timestamp operating mode
- `enum vtss_ts_internal_fmt_t { TS_INTERNAL_FMT_NONE, TS_INTERNAL_FMT_RESERVED_LEN_30BIT, TS_INTERNAL_FMT_RESERVED_LEN_32BIT, TS_INTERNAL_FMT_SUB_ADD_LEN_44BIT_CF62, TS_INTERNAL_FMT_RESERVED_LEN_48BIT_CF_3_0, TS_INTERNAL_FMT_RESERVED_LEN_48BIT_CF_0, TS_INTERNAL_FMT_RESERVED_LEN_48BIT_CF, TX_INTERNAL_FMT_MAX }`
parameter for setting the internal timestamp format

Functions

- `vtss_rc vtss_ts_timeofday_set (const vtss_inst_t inst, const vtss_timestamp_t *const ts)`
Set the current time in a Timestamp format.
- `vtss_rc vtss_ts_timeofday_set_delta (const vtss_inst_t inst, const vtss_timestamp_t *ts, BOOL negative)`
Set delta the current time in a Timestamp format.
- `vtss_rc vtss_ts_timeofday_offset_set (const vtss_inst_t inst, const i32 offset)`
Subtract offset from the current time.
- `vtss_rc vtss_ts_adjtimer_one_sec (const vtss_inst_t inst, BOOL *const ongoing_adjustment)`
Do the one sec administration in the Timestamp function.
- `vtss_rc vtss_ts_ongoing_adjustment (const vtss_inst_t inst, BOOL *const ongoing_adjustment)`
Check if the clock adjustment is ongoing.
- `vtss_rc vtss_ts_timeofday_get (const vtss_inst_t inst, vtss_timestamp_t *const ts, u32 *const tc)`
Get the current time in a Timestamp format, and the corresponding time counter.
- `vtss_rc vtss_ts_timeofday_next_pps_get (const vtss_inst_t inst, vtss_timestamp_t *const ts)`
Get the time at the next 1PPS pulse edge in a Timestamp format.
- `vtss_rc vtss_ts_adjtimer_set (const vtss_inst_t inst, const i32 adj)`
Adjust the clock timer ratio.
- `vtss_rc vtss_ts_adjtimer_get (const vtss_inst_t inst, i32 *const adj)`
get the clock timer ratio.
- `vtss_rc vtss_ts_freq_offset_get (const vtss_inst_t inst, i32 *const adj)`
get the clock internal timer frequency offset, compared to external clock input.
- `vtss_rc vtss_ts_external_clock_mode_get (const vtss_inst_t inst, vtss_ts_ext_clock_mode_t *const ext_clock_mode)`
Get the external clock mode. The mode depends on the hardware capability, it may be: Enable/disable external synch pulse Set clock output frequency.
- `vtss_rc vtss_ts_external_clock_mode_set (const vtss_inst_t inst, const vtss_ts_ext_clock_mode_t *const ext_clock_mode)`
Set the external clock mode. The mode depends on the hardware capability, it may be: Enable/disable external synch pulse Set clock output frequency.
- `vtss_rc vtss_ts_external_clock_saved_get (const vtss_inst_t inst, u32 *const saved)`
Get the latest saved time counter in nanosec.
- `vtss_rc vtss_ts_ingress_latency_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_timeinterval_t *const ingress_latency)`
Set the ingress latency.
- `vtss_rc vtss_ts_ingress_latency_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_timeinterval_t *const ingress_latency)`
Get the ingress latency.
- `vtss_rc vtss_ts_p2p_delay_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_timeinterval_t *const p2p_delay)`
Set the P2P delay.
- `vtss_rc vtss_ts_p2p_delay_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_timeinterval_t *const p2p_delay)`
Get the P2P delay.
- `vtss_rc vtss_ts_egress_latency_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_timeinterval_t *const egress_latency)`
Set the egress latency.
- `vtss_rc vtss_ts_egress_latency_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_timeinterval_t *const egress_latency)`
Get the egress latency.
- `vtss_rc vtss_ts_delay_asymmetry_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_timeinterval_t *const delay_asymmetry)`

- Set the delay asymmetry.*
- `vtss_rc vtss_ts_delay_asymmetry_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_timeinterval_t *const delay_asymmetry)`
 - Get the delay asymmetry.*
- `vtss_rc vtss_ts_operation_mode_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_ts_operation_mode_t *const mode)`
 - Set the timestamping operation mode for a port.*
- `vtss_rc vtss_ts_operation_mode_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_ts_operation_mode_t *const mode)`
 - Get the timestamping operation mode for a port.*
- `vtss_rc vtss_ts_internal_mode_set (const vtss_inst_t inst, const vtss_ts_internal_mode_t *const mode)`
 - Set the internal timestamping mode.*
- `vtss_rc vtss_ts_internal_mode_get (const vtss_inst_t inst, vtss_ts_internal_mode_t *const mode)`
 - Get the internal timestamping mode.*
- `vtss_rc vtss_tx_timestamp_update (const vtss_inst_t inst)`
 - Update the internal timestamp table, from HW.*
- `vtss_rc vtss_rx_timestamp_get (const vtss_inst_t inst, const vtss_ts_id_t *const ts_id, vtss_ts_timestamp_t *const ts)`
 - Get the rx FIFO timestamp for a {timestampId}.*
- `vtss_rc vtss_rx_timestamp_id_release (const vtss_inst_t inst, const vtss_ts_id_t *const ts_id)`
 - Release the FIFO rx timestamp id.*
- `vtss_rc vtss_rx_master_timestamp_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_ts_timestamp_t *const ts)`
 - Get rx timestamp from a port (convert from slave time to the master time)*
- `vtss_rc vtss_tx_timestamp_idx_alloc (const vtss_inst_t inst, const vtss_ts_timestamp_alloc_t *const alloc←_parm, vtss_ts_id_t *const ts_id)`
 - Allocate a timestamp id for a two step transmission.*
- `vtss_rc vtss_timestamp_age (const vtss_inst_t inst)`
 - Age the FIFO timestamps.*
- `vtss_rc vtss_ts_status_change (const vtss_inst_t inst, const vtss_port_no_t port_no)`
 - Signal port status change (used to detect and compensate for the internal ingress and egress latencies)*

8.40.1 Detailed Description

TimeStamping API.

This header file describes PTP/OAM TimeStamping API functions and associated types.

8.40.2 Function Documentation

8.40.2.1 `vtss_ts_timeofday_set()`

```
vtss_rc vtss_ts_timeofday_set (
    const vtss_inst_t inst,
    const vtss_timestamp_t *const ts )
```

Set the current time in a Timestamp format.

Parameters

<i>inst</i>	[IN] handle to an API instance.
<i>ts</i>	[IN] pointer to a TimeStamp structure.

Returns

Return code.

8.40.2.2 vtss_ts_timeofday_set_delta()

```
vtss_rc vtss_ts_timeofday_set_delta (
    const vtss_inst_t inst,
    const vtss_timestamp_t * ts,
    BOOL negative )
```

Set delta the current time in a Timestamp format.

Parameters

<i>inst</i>	[IN] handle to an API instance.
<i>ts</i>	[IN] pointer to a TimeStamp structure.
<i>negative</i>	[IN] True if ts is subtracted from current time, else ts is added.

Returns

Return code.

8.40.2.3 vtss_ts_timeofday_offset_set()

```
vtss_rc vtss_ts_timeofday_offset_set (
    const vtss_inst_t inst,
    const i32 offset )
```

Subtract offset from the current time.

Parameters

<i>inst</i>	[IN] handle to an API instance.
<i>offset</i>	[IN] offset in ns.

Returns

Return code.

8.40.2.4 vtss_ts_adjtimer_one_sec()

```
vtss_rc vtss_ts_adjtimer_one_sec (
    const vtss_inst_t inst,
    BOOL *const ongoing_adjustment )
```

Do the one sec administration in the Timestamp function.

Parameters

<i>inst</i>	[IN] handle to an API instance.
<i>ongoing_adjustment</i>	[OUT] True if clock adjustment is ongoing

Returns

Return code.

Jr1 : Maintains the clock setting process Caracal: Maintains the clock setting process Serval1: Maintains the clock setting process JR2 : it must only be called when the PPS output pin is low, therefore it shall be called at least 200 microseconds after the 1PPS interrupt

8.40.2.5 vtss_ts_ongoing_adjustment()

```
vtss_rc vtss_ts_ongoing_adjustment (
    const vtss_inst_t inst,
    BOOL *const ongoing_adjustment )
```

Check if the clock adjustment is ongoing.

Parameters

<i>inst</i>	[IN] handle to an API instance.
<i>ongoing_adjustment</i>	[OUT] True if clock adjustment is ongoing

Returns

Return code.

8.40.2.6 vtss_ts_timeofday_get()

```
vtss_rc vtss_ts_timeofday_get (
    const vtss_inst_t inst,
    vtss_timestamp_t *const ts,
    u32 *const tc )
```

Get the current time in a Timestamp format, and the corresponding time counter.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>ts</i>	[OUT] pointer to a TimeStamp structure
<i>tc</i>	[OUT] pointer to a time counter (internal hw format) Jaguar: tc = nanoseconds/nanosec_pr_clock_cycle (0..249999999) Caracal:tc = free running clock cycle counter Serval: tc = (nanoseconds + seconds*10**9) mod 2**32

Returns

Return code.

8.40.2.7 vtss_ts_timeofday_next_pps_get()

```
vtss_rc vtss_ts_timeofday_next_pps_get (
    const vtss_inst_t inst,
    vtss_timestamp_t *const ts )
```

Get the time at the next 1PPS pulse edge in a Timestamp format.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>ts</i>	[OUT] pointer to a TimeStamp structure

Returns

Return code.

8.40.2.8 vtss_ts_adjtimer_set()

```
vtss_rc vtss_ts_adjtimer_set (
    const vtss_inst_t inst,
    const i32 adj )
```

Adjust the clock timer ratio.

Parameters

<i>inst</i>	[IN] handle to an API instance.
<i>adj</i>	[IN] Clock ratio frequency offset in units of 0,1 ppb (parts pr billion). ratio > 0 => clock runs faster

Returns

Return code.

8.40.2.9 vtss_ts_adjtimer_get()

```
vtss_rc vtss_ts_adjtimer_get (
    const vtss_inst_t inst,
    i32 *const adj )
```

get the clock timer ratio.

Parameters

<i>inst</i>	[IN] handle to an API instance.
<i>adj</i>	[OUT] Clock ratio frequency offset in ppb (parts pr billion). ratio > 0 => clock runs faster

Returns

Return code.

8.40.2.10 vtss_ts_freq_offset_get()

```
vtss_rc vtss_ts_freq_offset_get (
    const vtss_inst_t inst,
    i32 *const adj )
```

get the clock internal timer frequency offset, compared to external clock input.

Parameters

<i>inst</i>	[IN] handle to an API instance.
<i>adj</i>	[OUT] Clock ratio frequency offset in ppb (parts pr billion). ratio > 0 => internal clock runs faster than external clock

Returns

Return code.

8.40.2.11 vtss_ts_external_clock_mode_get()

```
vtss_rc vtss_ts_external_clock_mode_get (
    const vtss_inst_t inst,
    vtss_ts_ext_clock_mode_t *const ext_clock_mode )
```

Get the external clock mode. The mode depends on the hardware capability, it may be: Enable/disable external synch pulse Set clock output frequency.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>ext_clock_mode</i>	[OUT] external clock mode.

Returns

Return code.

8.40.2.12 vtss_ts_external_clock_mode_set()

```
vtss_rc vtss_ts_external_clock_mode_set (
    const vtss_inst_t inst,
    const vtss_ts_ext_clock_mode_t *const ext_clock_mode )
```

Set the external clock mode. The mode depends on the hardware capability, it may be: Enable/disable external synch pulse Set clock output frequency.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>ext_clock_mode</i>	[IN] external clock mode.

Returns

Return code.

8.40.2.13 vtss_ts_external_clock_saved_get()

```
vtss_rc vtss_ts_external_clock_saved_get (
    const vtss_inst_t inst,
    u32 *const saved )
```

Get the latest saved time counter in nanosec.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>saved</i>	[OUT] latest saved value. [0..999.999.999]

Returns

Return code.

8.40.2.14 vtss_ts_ingress_latency_set()

```
vtss_rc vtss_ts_ingress_latency_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_timeinterval_t *const ingress_latency )
```

Set the ingress latency.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>ingress_latency</i>	[IN] pointer to ingress latency

Returns

Return code.

8.40.2.15 vtss_ts_ingress_latency_get()

```
vtss_rc vtss_ts_ingress_latency_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_timeinterval_t *const ingress_latency )
```

Get the ingress latency.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>ingress_latency</i>	[OUT] pointer to ingress_latency

Returns

Return code.

8.40.2.16 vtss_ts_p2p_delay_set()

```
vtss_rc vtss_ts_p2p_delay_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_timeinterval_t *const p2p_delay )
```

Set the P2P delay.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>p2p_delay</i>	[IN] peer-2-peer delay (measured)

Returns

Return code.

8.40.2.17 vtss_ts_p2p_delay_get()

```
vtss_rc vtss_ts_p2p_delay_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_timeinterval_t *const p2p_delay )
```

Get the P2P delay.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>p2p_delay</i>	[OUT] pointer to peer-2-peer delay

Returns

Return code.

8.40.2.18 vtss_ts_egress_latency_set()

```
vtss_rc vtss_ts_egress_latency_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_timeinterval_t *const egress_latency )
```

Set the egress latency.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>egress_latency</i>	[IN] egress latency

Returns

Return code.

8.40.2.19 vtss_ts_egress_latency_get()

```
vtss_rc vtss_ts_egress_latency_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_timeinterval_t *const egress_latency )
```

Get the egress latency.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>egress_latency</i>	[OUT] pointer to egress latency

Returns

Return code.

8.40.2.20 vtss_ts_delay_asymmetry_set()

```
vtss_rc vtss_ts_delay_asymmetry_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_timeinterval_t *const delay_asymmetry )
```

Set the delay asymmetry.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>delay_asymmetry</i>	[IN] delay asymmetry

Returns

Return code.

8.40.2.21 vtss_ts_delay_asymmetry_get()

```
vtss_rc vtss_ts_delay_asymmetry_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_timeinterval_t *const delay_asymmetry )
```

Get the delay asymmetry.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>delay_asymmetry</i>	[OUT] pointer to delay asymmetry

Returns

Return code.

8.40.2.22 vtss_ts_operation_mode_set()

```
vtss_rc vtss_ts_operation_mode_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_ts_operation_mode_t *const mode )
```

Set the timestamping operation mode for a port.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>mode</i>	[IN] pointer to a struct holding the operation mode

Returns

Return code.

Serval: Used to set backplane (INTERNAL) mode/normal(EXTERNAL) mode Other : Not used

8.40.2.23 vtss_ts_operation_mode_get()

```
vtss_rc vtss_ts_operation_mode_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_ts_operation_mode_t *const mode )
```

Get the timestamping operation mode for a port.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>mode</i>	[OUT] pointer to a struct holding the operation mode

Returns

Return code.

8.40.2.24 vtss_ts_internal_mode_set()

```
vtss_rc vtss_ts_internal_mode_set (
    const vtss_inst_t inst,
    const vtss_ts_internal_mode_t *const mode )
```

Set the internal timestamping mode.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>mode</i>	[IN] pointer to a struct holding the operation mode

Returns

Return code.

Serval: Used to set INTERNAL mode timestamping format Other : Not used

8.40.2.25 vtss_ts_internal_mode_get()

```
vtss_rc vtss_ts_internal_mode_get (
    const vtss_inst_t inst,
    vtss_ts_internal_mode_t *const mode )
```

Get the internal timestamping mode.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>mode</i>	[OUT] pointer to a struct holding the operation mode

Returns

Return code.

8.40.2.26 vtss_tx_timestamp_update()

```
vtss_rc vtss_tx_timestamp_update (
    const vtss_inst_t inst )
```

Update the internal timestamp table, from HW.

Parameters

<i>inst</i>	[IN] handle to an API instance
-------------	--------------------------------

Returns

Return code.

8.40.2.27 vtss_rx_timestamp_get()

```
vtss_rc vtss_rx_timestamp_get (
    const vtss_inst_t inst,
    const vtss_ts_id_t *const ts_id,
    vtss_ts_timestamp_t *const ts )
```

Get the rx FIFO timestamp for a {timestampId}.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>ts_id</i>	[IN] timestamp id
<i>ts</i>	[OUT] pointer to a struct holding the fifo timestamp

Returns

Return code.

8.40.2.28 vtss_rx_timestamp_id_release()

```
vtss_rc vtss_rx_timestamp_id_release (
    const vtss_inst_t inst,
    const vtss_ts_id_t *const ts_id )
```

Release the FIFO rx timestamp id.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>ts_id</i>	[IN] timestamp id

Returns

Return code.

8.40.2.29 vtss_rx_master_timestamp_get()

```
vtss_rc vtss_rx_master_timestamp_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_ts_timestamp_t *const ts )
```

Get rx timestamp from a port (convert from slave time to the master time)

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>ts</i>	[IN/OUT] pointer to a struct holding the timestamp

Returns

Return code.

8.40.2.30 vtss_tx_timestamp_idx_alloc()

```
vtss_rc vtss_tx_timestamp_idx_alloc (
    const vtss_inst_t inst,
    const vtss_ts_timestamp_alloc_t *const alloc_parm,
    vtss_ts_id_t *const ts_id )
```

Allocate a timestamp id for a two step transmission.

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>alloc_parm</i>	[IN] pointer allocation parameters
<i>ts_id</i>	[OUT] timestamp id

Returns

Return code.

8.40.2.31 vtss_timestamp_age()

```
vtss_rc vtss_timestamp_age (
    const vtss_inst_t inst )
```

Age the FIFO timestamps.

Parameters

<i>inst</i>	[IN] handle to an API instance
-------------	--------------------------------

Returns

Return code.

8.40.2.32 vtss_ts_status_change()

```
vtss_rc vtss_ts_status_change (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no )
```

Signal port status change (used to detect and compensate for the internal ingress and egress latencies)

Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number

Returns

Return code.

8.41 vtss_api/include/vtss_upi_api.h File Reference

Define UPI API interface.

```
#include <vtss/api/types.h>
```

8.41.1 Detailed Description

Define UPI API interface.

8.42 vtss_api/include/vtss_wis_api.h File Reference

eWIS layer API

```
#include <vtss/api/types.h>
```

Data Structures

- struct [vtss_ewis_tti_s](#)
Trail Trace Identifier type.
- struct [vtss_ewis_fault_cons_act_s](#)
eWIS fault mask configuration, i.e set up which defects trigger the Fault condition
- struct [vtss_ewis_aisl_cons_act_s](#)
eWIS AIS-L consequent actions
- struct [vtss_ewis_rdil_cons_act_s](#)
eWIS RDI-L consequent actions
- struct [vtss_ewis_cons_act_s](#)
eWIS consequent actions
- struct [vtss_ewis_line_force_mode_s](#)
eWIS line force mode
- struct [vtss_ewis_line_tx_force_mode_s](#)
eWIS line TX force mode
- struct [vtss_ewis_path_force_mode_s](#)
eWIS path force modes
- struct [vtss_ewis_force_mode_s](#)
eWIS force modes
- struct [vtss_ewis_perf_mode_s](#)
eWIS Mode(Bit/Block) for the Performance Monitoring Counters
- struct [vtss_ewis_status_s](#)
eWIS status
- struct [vtss_ewis_defects_s](#)
eWIS defects
- struct [vtss_ewis_perf_s](#)
eWIS performance primitives. These data are assumed to be read once every sec. The counters holds increments compared to previous read. The namings and definitions are taken from ITU-T rec G.783.
- struct [vtss_ewis_counter_s](#)

- *eWIS performance counters. These counters are free running counters that wraps to zero.*
- struct [vtss_ewis_test_conf_s](#)
eWIS test configuration
- struct [vtss_ewis_test_status_s](#)
eWIS test status
- struct [vtss_ewis_tx_oh_s](#)
WIS transmitted overhead data. only a few oh bytes can be set dynamically. These OH bytes are not configurable from the API: H4 : multiframe indicator M0/M1: STS-1 Line Remote Error Indication (REI) G1: Path status.
- struct [vtss_ewis_tx_passthru_s](#)
eWIS overhead passthru configuration.
- struct [vtss_ewis_counter_threshold_s](#)
eWIS performance counter thresholds.
- struct [vtss_ewis_static_conf_s](#)
eWIS static configuration data,
- struct [vtss_ewis_sl_conf_s](#)
signal label configuration
- struct [vtss_ewis_conf_s](#)
eWIS configuration primitives, used to hold all the configuration parameters in the internal state in the API.

Macros

- #define [VTSS_EWIS_SEF_EV](#) 0x00000001
WIS interrupt events.
- #define [VTSS_EWIS_FPLM_EV](#) 0x00000002
- #define [VTSS_EWIS_FAIS_EV](#) 0x00000004
- #define [VTSS_EWIS_LOF_EV](#) 0x00000008
- #define [VTSS_EWIS_LOS_EV](#) 0x00000010
- #define [VTSS_EWIS_RDIL_EV](#) 0x00000020
- #define [VTSS_EWIS_AISL_EV](#) 0x00000040
- #define [VTSS_EWIS_LCDP_EV](#) 0x00000080
- #define [VTSS_EWIS_PLMP_EV](#) 0x00000100
- #define [VTSS_EWIS_AISP_EV](#) 0x00000200
- #define [VTSS_EWIS_LOPP_EV](#) 0x00000400
- #define [VTSS_EWIS_MODULE_EV](#) 0x00000800
- #define [VTSS_EWIS_TXLOL_EV](#) 0x00001000
- #define [VTSS_EWIS_RXLOL_EV](#) 0x00002000
- #define [VTSS_EWIS_LOPC_EV](#) 0x00004000
- #define [VTSS_EWIS_UNEQP_EV](#) 0x00008000
- #define [VTSS_EWIS_FEUNEQP_EV](#) 0x00010000
- #define [VTSS_EWIS_FERDIP_EV](#) 0x00020000
- #define [VTSS_EWIS_REIL_EV](#) 0x00040000
- #define [VTSS_EWIS_REIP_EV](#) 0x00080000
- #define [VTSS_EWIS_HIGH_BER_EV](#) 0x00100000
- #define [VTSS_EWIS_PCS_RECEIVE_FAULT_PEND](#) 0x00200000
- #define [VTSS_EWIS_B1_NZ_EV](#) 0x00400000
- #define [VTSS_EWIS_B2_NZ_EV](#) 0x00800000
- #define [VTSS_EWIS_B3_NZ_EV](#) 0x01000000
- #define [VTSS_EWIS_REIL_NZ_EV](#) 0x02000000
- #define [VTSS_EWIS_REIP_NZ_EV](#) 0x04000000
- #define [VTSS_EWIS_B1_THRESH_EV](#) 0x08000000
- #define [VTSS_EWIS_B2_THRESH_EV](#) 0x10000000
- #define [VTSS_EWIS_B3_THRESH_EV](#) 0x20000000
- #define [VTSS_EWIS_REIL_THRESH_EV](#) 0x40000000
- #define [VTSS_EWIS_REIP_THRESH_EV](#) 0x80000000

Typedefs

- `typedef struct vtss_ewis_tti_s vtss_ewis_tti_t`
Trace Identifier type.
- `typedef struct vtss_ewis_fault_cons_act_s vtss_ewis_fault_cons_act_t`
eWIS fault mask configuration, i.e set up which defects trigger the Fault condition
- `typedef struct vtss_ewis_aisl_cons_act_s vtss_ewis_aisl_cons_act_t`
eWIS AIS-L consequent actions
- `typedef struct vtss_ewis_rdil_cons_act_s vtss_ewis_rdil_cons_act_t`
eWIS RDI-L consequent actions
- `typedef struct vtss_ewis_cons_act_s vtss_ewis_cons_act_t`
eWIS consequent actions
- `typedef struct vtss_ewis_line_force_mode_s vtss_ewis_line_force_mode_t`
eWIS line force mode
- `typedef struct vtss_ewis_line_tx_force_mode_s vtss_ewis_line_tx_force_mode_t`
eWIS line TX force mode
- `typedef struct vtss_ewis_path_force_mode_s vtss_ewis_path_force_mode_t`
eWIS path force modes
- `typedef struct vtss_ewis_force_mode_s vtss_ewis_force_mode_t`
eWIS force modes
- `typedef struct vtss_ewis_perf_mode_s vtss_ewis_perf_mode_t`
eWIS Mode(Bit/Block) for the Performance Monitoring Counters
- `typedef struct vtss_ewis_status_s vtss_ewis_status_t`
eWIS status
- `typedef struct vtss_ewis_defects_s vtss_ewis_defects_t`
eWIS defects
- `typedef struct vtss_ewis_perf_s vtss_ewis_perf_t`
eWIS performance primitives. These data are assumed to be read once every sec. The counters holds increments compared to previous read. The namings and definitions are taken from ITU-T rec G.783.
- `typedef struct vtss_ewis_counter_s vtss_ewis_counter_t`
eWIS performance counters. These counters are free running counters that wraps to zero.
- `typedef enum vtss_ewis_test_pattern_s vtss_ewis_test_pattern_t`
eWIS test pattern mode types.
- `typedef struct vtss_ewis_test_conf_s vtss_ewis_test_conf_t`
eWIS test configuration
- `typedef struct vtss_ewis_test_status_s vtss_ewis_test_status_t`
eWIS test status
- `typedef struct vtss_ewis_tx_oh_s vtss_ewis_tx_oh_t`
WIS transmitted overhead data. only a few oh bytes can be set dynamically. These OH bytes are not configurable from the API: H4 : multiframe indicator M0/M1: STS-1 Line Remote Error Indication (REI) G1: Path status.
- `typedef struct vtss_ewis_tx_passthru_s vtss_ewis_tx_oh_passthru_t`
eWIS overhead passthru configuration.
- `typedef struct vtss_ewis_counter_threshold_s vtss_ewis_counter_threshold_t`
eWIS performance counter thresholds.
- `typedef struct vtss_ewis_static_conf_s vtss_ewis_static_conf_t`
eWIS static configuration data,
- `typedef struct vtss_ewis_sl_conf_s vtss_ewis_sl_conf_t`
signal label configuration
- `typedef struct vtss_ewis_conf_s vtss_ewis_conf_t`
eWIS configuration primitives, used to hold all the configuration parameters in the internal state in the API.
- `typedef u64 vtss_ewis_event_t`

Enumerations

- enum `vtss_ewis_tti_mode_t` { `TTI_MODE_1`, `TTI_MODE_16`, `TTI_MODE_64`, `TTI_MODE_MAX` }

Trace Identifier mode types.
- enum `vtss_ewis_perf_cntr_mode_t` { `VTSS_EWIS_PERF_MODE_BIT`, `VTSS_EWIS_PERF_MODE_BLOCK` }

eWIS Mode(Bit/Block) for the Performance Monitoring Counters
- enum `vtss_ewis_mode_t` {
 `VTSS_WIS_OPERMODE_DISABLE`, `VTSS_WIS_OPERMODE_WIS_MODE`, `VTSS_WIS_OPERMODE_STS192`,
`VTSS_WIS_OPERMODE_STM64`,
`VTSS_WIS_OPERMODE_MAX` }

eWIS operational mode types
- enum `vtss_ewis_test_pattern_s` {
 `VTSS_WIS_TEST_MODE_DISABLE`, `VTSS_WIS_TEST_MODE_SQUARE_WAVE`, `VTSS_WIS_TEST_MODE_PRBS31`,
`VTSS_WIS_TEST_MODE_MIXED_FREQUENCY`,
`VTSS_WIS_TEST_MODE_MAX` }

eWIS test pattern mode types.
- enum `vtss_ewis_prbs31_err_inj_t` { `EWIS_PRBS31_SINGLE_ERR`, `EWIS_PRBS31_SAT_ERR`, `EWIS_P←RBS31_MODE_MAX` }

test error injection types

Functions

- `vtss_rc vtss_ewis_event_enable` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `BOOL` enable, const `vtss_ewis_event_t` ev_mask)

Enable event generation for a specific event type or group of events.
- `vtss_rc vtss_ewis_event_poll` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_ewis_event_t` *const status)

Polling function called at by interrupt or periodically.
- `vtss_rc vtss_ewis_event_poll_without_mask` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_ewis_event_t` *const status)

Polling function called at by interrupt or periodically.
- `vtss_rc vtss_ewis_event_force` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `BOOL` enable, const `vtss_ewis_event_t` ev_force)

Forces one or more WIS events to occur (simulated events)
- `vtss_rc vtss_ewis_static_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_ewis_static_conf_t` *const stat_conf)

Get eWIS static configuration.
- `vtss_rc vtss_ewis_force_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_ewis_force_mode_t` *const force_conf)

Set WIS force mode configuration. The WIS can be forced to induce a particular condition, These can be configured in `vtss_ewis_force_mode_t`. AIS,RDI can be forced in Tx and Rx directions and force UNEQ and RDI in path layer.
- `vtss_rc vtss_ewis_force_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_ewis_force_mode_t` *const force_conf)

Get WIS force mode configuration.
- `vtss_rc vtss_ewis_tx_oh_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_ewis_tx_oh_t` *const tx_oh)

Set WIS transmitted overhead bytes. Supports insertion of various Section, line and path overhead Bytes.
- `vtss_rc vtss_ewis_tx_oh_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_ewis_tx_oh_t` *const tx_oh)

Get configured WIS transmitted overhead bytes.
- `vtss_rc vtss_ewis_tx_oh_passthru_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_ewis_tx_oh_passthru_t` *const tx_oh_passthru)

Set WIS transmitted overhead bytes.

- `vtss_rc vtss_ewis_tx_oh_passthru_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_ewis_tx_oh_passthru_t` *const tx_oh_passthru)

Set eWIS overhead passthru configuration.
- `vtss_rc vtss_ewis_mode_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_ewis_mode_t` *const mode)

Get eWIS overhead passthru configuration.
- `vtss_rc vtss_ewis_mode_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_ewis_mode_t` *const mode)

Set eWIS mode.
- `vtss_rc vtss_ewis_reset` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no)

Reset WIS block.
- `vtss_rc vtss_ewis_cons_act_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_ewis_cons_act_t` *const cons_act)

Set consequent actions, i.e. how to handle AIS-L insertion and RDI-L backreporting.
- `vtss_rc vtss_ewis_cons_act_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_ewis_cons_act_t` *const cons_act)

Get the configured consequent actions.
- `vtss_rc vtss_ewis_section_ttxi_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_ewis_tti_t` *const ttxi)

Set section transmitted Trail Trace Identifier. The transmitted trace identifier is aligned according to the specification, i.e 16 byte mode: In the first byte MSB = 1, and all other bytes MSB = 0. 64 byte mode: this is a text string terminated with CR/LF.
- `vtss_rc vtss_ewis_section_ttxi_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_ewis_tti_t` *const ttxi)

Get the configured section transmitted Trail Trace Identifier.
- `vtss_rc vtss_ewis_exp_sl_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_ewis_sl_conf_t` *const sl)

Set expected Signal label. The signal label is only configurable in SONET/SDH mode, in this mode the path overhead is not terminated, it is only monitored, i.e. only expected signal label is configurable.
- `vtss_rc vtss_ewis_path_ttxi_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_ewis_tti_t` *const ttxi)

*Set Path Transmitted Trail Trace Identifier. The transmitted trace identifier is aligned according to the specification, i.e 16 byte mode: In the first byte MSB = 1, and all other bytes MSB = 0. 64 byte mode: this is a text string terminated with CR/LF. **
- `vtss_rc vtss_ewis_path_ttxi_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_ewis_tti_t` *const ttxi)

Get the configured Path Transmitted Trail Trace Identifier.
- `vtss_rc vtss_ewis_test_mode_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_ewis_test_conf_t` *const test_mode)

Set WIS test mode.
- `vtss_rc vtss_ewis_test_mode_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_ewis_test_conf_t` *const test_mode)

Get eWIS test mode.
- `vtss_rc vtss_ewis_prbs31_err_inj_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_ewis_prbs31_err_inj_t` *const inj)

Inject eWIS PRBS31 errors.
- `vtss_rc vtss_ewis_test_counter_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_ewis_test_status_t` *const test_status)

Get eWIS test counter.
- `vtss_rc vtss_ewis_defects_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_ewis_defects_t` *const def)

Get eWIS defects. Reports the current status of the defects.

- `vtss_rc vtss_ewis_status_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_ewis_status_t` *const status)
Get eWIS fault and link status.
- `vtss_rc vtss_ewis_section_acti_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_ewis_tti_t` *const acti)
Get section received (accepted) Trail Trace Identifier. The received trace identifier is aligned according to the specification, i.e 16 byte mode: In the first byte MSB = 1, and all other bytes MSB = 0. (see G.707 section 9.2.2.2) No CRC checksum verification is done in the API. 64 byte mode: this is a text string terminated with CR/LF.
- `vtss_rc vtss_ewis_path_acti_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_ewis_tti_t` *const acti)
Get path received (accepted) Trail Trace Identifier.
- `vtss_rc vtss_ewis_counter_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_ewis_counter_t` *const counter)
Get free running eWIS counters.
- `vtss_rc vtss_ewis_perf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_ewis_perf_t` *const perf)
Get eWIS counters per second (performance primitives). By default the source of PMTICK event (generation of 1 second) is configured to be internal. The values are accumulate for a period of 1 second and are then updated in the associated registers.
- `vtss_rc vtss_ewis_counter_threshold_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_ewis_counter_threshold_t` *const threshold)
Set eWIS error counter thresholds per second. The PHY generates an interrupt once the error counter exceeds the configured threshold values. The units is frames per second. The threshold value configuration is possible for B1,B2,B3,REIP,REIL errors. Eg: a threshold value of 0 gives one interrupt for every 1 second if the error counter is atleast 1.
- `vtss_rc vtss_ewis_counter_threshold_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_ewis_counter_threshold_t` *const threshold)
Get the configured eWIS error counter thresholds.
- `vtss_rc vtss_ewis_perf_mode_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_ewis_perf_mode_t` const *perf_mode)
Set the eWIS performance block counter modes.
- `vtss_rc vtss_ewis_perf_mode_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_ewis_perf_mode_t` *const perf_mode)
Get the eWIS performance block counter modes.

8.42.1 Detailed Description

eWIS layer API

8.42.2 Macro Definition Documentation

8.42.2.1 VTSS_EWIS_SEF_EV

```
#define VTSS_EWIS_SEF_EV 0x00000001
```

WIS interrupt events.

Note

These interrupts are not used for 8487-15/8488-15. There are separate type `vtss_phy_10g_event_t` defined in `vtss_phy_10g_api.h` for these chips. SEF has changed state

Definition at line 338 of file `vtss_wis_api.h`.

8.42.2.2 VTSS_EWIS_FPLM_EV

```
#define VTSS_EWIS_FPLM_EV 0x00000002
```

far-end (PLM-P) / (LCDP)

Definition at line 339 of file vtss_wis_api.h.

8.42.2.3 VTSS_EWIS_FAIS_EV

```
#define VTSS_EWIS_FAIS_EV 0x00000004
```

far-end (AIS-P) / (LOP)

Definition at line 340 of file vtss_wis_api.h.

8.42.2.4 VTSS_EWIS_LOF_EV

```
#define VTSS_EWIS_LOF_EV 0x00000008
```

Loss of Frame (LOF)

Definition at line 341 of file vtss_wis_api.h.

8.42.2.5 VTSS_EWIS_LOS_EV

```
#define VTSS_EWIS_LOS_EV 0x00000010
```

Loss of Signal (LOS)

Definition at line 342 of file vtss_wis_api.h.

8.42.2.6 VTSS_EWIS_RDIL_EV

```
#define VTSS_EWIS_RDIL_EV 0x00000020
```

Line Remote Defect Indication (RDI-L)

Definition at line 343 of file vtss_wis_api.h.

8.42.2.7 VTSS_EWIS_AISL_EV

```
#define VTSS_EWIS_AISL_EV 0x00000040
```

Line Alarm Indication Signal (AIS-L)

Definition at line 344 of file vtss_wis_api.h.

8.42.2.8 VTSS_EWIS_LCDP_EV

```
#define VTSS_EWIS_LCDP_EV 0x00000080
```

Loss of Code-group Delineation (LCD-P)

Definition at line 345 of file vtss_wis_api.h.

8.42.2.9 VTSS_EWIS_PLMP_EV

```
#define VTSS_EWIS_PLMP_EV 0x00000100
```

Path Label Mismatch (PLMP)

Definition at line 346 of file vtss_wis_api.h.

8.42.2.10 VTSS_EWIS_AISP_EV

```
#define VTSS_EWIS_AISP_EV 0x00000200
```

Path Alarm Indication Signal (AIS-P)

Definition at line 347 of file vtss_wis_api.h.

8.42.2.11 VTSS_EWIS_LOPP_EV

```
#define VTSS_EWIS_LOPP_EV 0x00000400
```

Path Loss of Pointer (LOP-P)

Definition at line 348 of file vtss_wis_api.h.

8.42.2.12 VTSS_EWIS_MODULE_EV

```
#define VTSS_EWIS_MODULE_EV 0x00000800
```

GPIO pin state being driven by optics module

Definition at line 349 of file vtss_wis_api.h.

8.42.2.13 VTSS_EWIS_TXLOL_EV

```
#define VTSS_EWIS_TXLOL_EV 0x00001000
```

PMA CMU Loss of Lock

Definition at line 350 of file vtss_wis_api.h.

8.42.2.14 VTSS_EWIS_RXLOL_EV

```
#define VTSS_EWIS_RXLOL_EV 0x00002000
```

PMA CRU Loss of Lock

Definition at line 351 of file vtss_wis_api.h.

8.42.2.15 VTSS_EWIS_LOPC_EV

```
#define VTSS_EWIS_LOPC_EV 0x00004000
```

Loss of Optical Carrier (LOPC)

Definition at line 352 of file vtss_wis_api.h.

8.42.2.16 VTSS_EWIS_UNEQP_EV

```
#define VTSS_EWIS_UNEQP_EV 0x00008000
```

Unequiped Path (UNEQ-P)

Definition at line 353 of file vtss_wis_api.h.

8.42.2.17 VTSS_EWIS_FEUNEQP_EV

```
#define VTSS_EWIS_FEUNEQP_EV 0x00010000
```

Far-end Unequiped Path (UNEQ-P)

Definition at line 354 of file vtss_wis_api.h.

8.42.2.18 VTSS_EWIS_FERDIP_EV

```
#define VTSS_EWIS_FERDIP_EV 0x00020000
```

Far-end Path Remote Defect Identifier (RDI-P)

Definition at line 355 of file vtss_wis_api.h.

8.42.2.19 VTSS_EWIS_REIL_EV

```
#define VTSS_EWIS_REIL_EV 0x00040000
```

Line Remote Error Indication (REI-L)

Definition at line 356 of file vtss_wis_api.h.

8.42.2.20 VTSS_EWIS_REIP_EV

```
#define VTSS_EWIS_REIP_EV 0x00080000
```

Path Remote Error Indication (REI-P)

Definition at line 357 of file vtss_wis_api.h.

8.42.2.21 VTSS_EWIS_HIGH_BER_EV

```
#define VTSS_EWIS_HIGH_BER_EV 0x00100000
```

PCS high bit error rate (BER)

Definition at line 358 of file vtss_wis_api.h.

8.42.2.22 VTSS_EWIS_PCS_RECEIVE_FAULT_PEND

```
#define VTSS_EWIS_PCS_RECEIVE_FAULT_PEND 0x00200000
```

PCS Receive fault

Definition at line 360 of file vtss_wis_api.h.

8.42.2.23 VTSS_EWIS_B1_NZ_EV

```
#define VTSS_EWIS_B1_NZ_EV 0x00400000
```

PMTICK B1 BIP (B1_ERR_CNT) not zero

Definition at line 362 of file vtss_wis_api.h.

8.42.2.24 VTSS_EWIS_B2_NZ_EV

```
#define VTSS_EWIS_B2_NZ_EV 0x00800000
```

PMTICK B2 BIP (B1_ERR_CNT) not zero

Definition at line 363 of file vtss_wis_api.h.

8.42.2.25 VTSS_EWIS_B3_NZ_EV

```
#define VTSS_EWIS_B3_NZ_EV 0x01000000
```

PMTICK B3 BIP (B1_ERR_CNT) not zero

Definition at line 364 of file vtss_wis_api.h.

8.42.2.26 VTSS_EWIS_REIL_NZ_EV

```
#define VTSS_EWIS_REIL_NZ_EV 0x02000000
```

PMTICK REI-L (REIL_ERR_CNT) not zero

Definition at line 365 of file vtss_wis_api.h.

8.42.2.27 VTSS_EWIS_REIP_NZ_EV

```
#define VTSS_EWIS_REIP_NZ_EV 0x04000000
```

PMTICK REI-P (REIP_ERR_CNT) not zero

Definition at line 366 of file vtss_wis_api.h.

8.42.2.28 VTSS_EWIS_B1_THRESH_EV

```
#define VTSS_EWIS_B1_THRESH_EV 0x08000000
```

B1_THRESH_ERR

Definition at line 368 of file vtss_wis_api.h.

8.42.2.29 VTSS_EWIS_B2_THRESH_EV

```
#define VTSS_EWIS_B2_THRESH_EV 0x10000000
```

B2_THRESH_ERR

Definition at line 369 of file vtss_wis_api.h.

8.42.2.30 VTSS_EWIS_B3_THRESH_EV

```
#define VTSS_EWIS_B3_THRESH_EV 0x20000000
```

B3_THRESH_ERR

Definition at line 370 of file vtss_wis_api.h.

8.42.2.31 VTSS_EWIS_REIL_THRESH_EV

```
#define VTSS_EWIS_REIL_THRESH_EV 0x40000000
```

REIL_THRESH_ERR

Definition at line 371 of file vtss_wis_api.h.

8.42.2.32 VTSS_EWIS_REIP_THRESH_EV

```
#define VTSS_EWIS_REIP_THRESH_EV 0x80000000
```

REIP_THRESH_ERR

Definition at line 372 of file vtss_wis_api.h.

8.42.3 Typedef Documentation

8.42.3.1 vtss_ewis_static_conf_t

```
typedef struct vtss_ewis_static_conf_s vtss_ewis_static_conf_t
```

eWIS static configuration data,

Note

This is specific to 8487/8488-15 and should not be used for Daytona.

8.42.3.2 vtss_ewis_event_t

```
typedef u64 vtss_ewis_event_t
```

Int events: Single event or 'OR' multiple events above

Definition at line 396 of file vtss_wis_api.h.

8.42.4 Enumeration Type Documentation

8.42.4.1 vtss_ewis_tti_mode_t

```
enum vtss_ewis_tti_mode_t
```

Trail Trace Identifier mode types.

Enumerator

TTI_MODE_1	one byte trace identifier
TTI_MODE_16	16 bytes trace identifier
TTI_MODE_64	64 bytes trace identifier

Definition at line 47 of file vtss_wis_api.h.

8.42.4.2 vtss_ewis_perf_cntr_mode_t

```
enum vtss_ewis_perf_cntr_mode_t
```

eWIS Mode(Bit/Block) for the Performance Monitoring Counters

Enumerator

VTSS_EWIS_PERF_MODE_BIT	Bit mode of the perf monitor counter
VTSS_EWIS_PERF_MODE_BLOCK	Block mode of the perf monitor counter

Definition at line 123 of file vtss_wis_api.h.

8.42.4.3 vtss_ewis_mode_t

```
enum vtss_ewis_mode_t
```

eWIS operational mode types

Enumerator

VTSS_WIS_OPERMODE_WIS_MODE	WIS mode disabled
VTSS_WIS_OPERMODE_STS192	WIS mode enabled
VTSS_WIS_OPERMODE_STM64	WIS mode SONET - STS192
VTSS_WIS_OPERMODE_MAX	WIS mode SDH - STM64 WIS mode Invalid

Definition at line 138 of file vtss_wis_api.h.

8.42.4.4 vtss_ewis_test_pattern_s

```
enum vtss_ewis_test_pattern_s
```

eWIS test pattern mode types.

Enumerator

VTSS_WIS_TEST_MODE_DISABLE	Disable test
VTSS_WIS_TEST_MODE_SQUARE_WAVE	Enable squarewave generator, Only valid for test generator
VTSS_WIS_TEST_MODE_PRBS31	Enable prbs31 generator / analyzer (not supported in Daytona)
VTSS_WIS_TEST_MODE_MIXED_FREQUENCY	Enable mixed frequency generator / analyzer
Generated by Doxygen	Test mode Invalid

Definition at line 197 of file vtss_wis_api.h.

8.42.4.5 vtss_ewis_prbs31_err_inj_t

enum `vtss_ewis_prbs31_err_inj_t`

test error injection types

Enumerator

<code>EWIS_PRBS31_SINGLE_ERR</code>	Inject a single bit error (=> error counter incrementing by 3)
<code>EWIS_PRBS31_SAT_ERR</code>	Force the PRBS31 pattern error counter to a value of 65528 (close to saturation)

Definition at line 278 of file vtss_wis_api.h.

8.42.5 Function Documentation

8.42.5.1 vtss_ewis_event_enable()

```
vtss_rc vtss_ewis_event_enable (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const BOOL enable,
    const vtss_ewis_event_t ev_mask )
```

Enable event generation for a specific event type or group of events.

Note

Not applicable for 8487/8488-15

Parameters

<code>inst</code>	[IN] Target instance reference.
<code>port_no</code>	[IN] Port number
<code>enable</code>	[IN] Enable or disable events
<code>ev_mask</code>	[IN] Event type(s) to control (mask)

Returns

Return code.

8.42.5.2 vtss_ewis_event_poll()

```
vtss_rc vtss_ewis_event_poll (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_ewis_event_t *const status )
```

Polling function called at by interrupt or periodically.

Note

Interrupt status will be cleared on read. Not applicable for 8487/8488-15

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>status</i>	[OUT] Event status, bit set indicates corresponding event/interrupt has detected

Returns

Return code.

8.42.5.3 vtss_ewis_event_poll_without_mask()

```
vtss_rc vtss_ewis_event_poll_without_mask (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_ewis_event_t *const status )
```

Polling function called at by interrupt or periodically.

Note

Interrupt status will be cleared on read. Not applicable for 8487/8488-15

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>status</i>	[OUT] Event status, bit set indicates corresponding event/interrupt has detected irrespective of the mask register

Returns

Return code.

8.42.5.4 vtss_ewis_event_force()

```
vtss_rc vtss_ewis_event_force (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const BOOL enable,
    const vtss_ewis_event_t ev_force )
```

Forces one or more WIS events to occur (simulated events)

Note

useful in debugging.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>enable</i>	[IN] Enable or disable events
<i>ev_force</i>	[IN] Mask defining which events are forces

Returns

Return code.

8.42.5.5 vtss_ewis_static_conf_get()

```
vtss_rc vtss_ewis_static_conf_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_ewis_static_conf_t *const stat_conf )
```

Get eWIS static configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>stat_conf</i>	[OUT] Get eWIS Static configuration, i.e configuration that is set up at initialization, and not changed afterwards.

Returns

Return code.

8.42.5.6 vtss_ewis_force_conf_set()

```
vtss_rc vtss_ewis_force_conf_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_ewis_force_mode_t *const force_conf )
```

Set WIS force mode configuration. The WIS can be forced to induce a particular condition, These can be configured in vtss_ewis_force_mode_t. AIS,RDI can be forced in Tx and Rx directions and force UNEQ and RDI in path layer.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>force_conf</i>	[IN] Set force mode.

Returns

Return code.

8.42.5.7 vtss_ewis_force_conf_get()

```
vtss_rc vtss_ewis_force_conf_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_ewis_force_mode_t *const force_conf )
```

Get WIS force mode configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>force_conf</i>	[OUT] Get force mode configuration.

Returns

Return code.

8.42.5.8 vtss_ewis_tx_oh_set()

```
vtss_rc vtss_ewis_tx_oh_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_ewis_tx_oh_t *const tx_oh )
```

Set WIS transmitted overhead bytes. Supports insertion of various Section, line and path overhead Bytes.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>tx_oh</i>	[IN] Transmitted overhead byte values

Returns

Return code.

8.42.5.9 vtss_ewis_tx_oh_get()

```
vtss_rc vtss_ewis_tx_oh_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_ewis_tx_oh_t *const tx_oh )
```

Get configured WIS transmitted overhead bytes.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>tx_oh</i>	[OUT] Transmitted overhead byte values

Returns

Return code.

8.42.5.10 vtss_ewis_tx_oh_passthru_set()

```
vtss_rc vtss_ewis_tx_oh_passthru_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_ewis_tx_oh_passthru_t *const tx_oh_passthru )
```

Set eWIS overhead passthru configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>tx_oh_passthru</i>	[IN] Transmitted overhead passthrough configuration

Returns

Return code.

8.42.5.11 vtss_ewis_tx_oh_passthru_get()

```
vtss_rc vtss_ewis_tx_oh_passthru_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_ewis_tx_oh_passthru_t *const tx_oh_passthru )
```

Get eWIS overhead passthru configuration.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>tx_oh_passthru</i>	[OUT] Transmitted overhead passthrough configuration

Returns

Return code.

8.42.5.12 vtss_ewis_mode_set()

```
vtss_rc vtss_ewis_mode_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_ewis_mode_t *const mode )
```

Set eWIS mode.

Note

Should not used for 8487-15/8488-15. The mode configuration is enabled by calling vtss_phy_10g_mode_set in the case of 8487-15. In Daytona this is useful in setting the WIS block to operate in multiple modes.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>mode</i>	[IN] Set WIS mode (Disable, WIS, STS192, STM64). sts192 (full Sonet/SDH termination is only supported in Daytona)

Returns

Return code.

8.42.5.13 vtss_ewis_mode_get()

```
vtss_rc vtss_ewis_mode_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_ewis_mode_t *const mode )
```

Get WIS mode.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>mode</i>	[OUT] Get WIS mode (Disable, WIS, STS192, STM64).

Returns

Return code.

8.42.5.14 vtss_ewis_reset()

```
vtss_rc vtss_ewis_reset (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no )
```

Reset WIS block.

Note

Useful only for 8487-17/8488-15.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.

Returns

Return code.

8.42.5.15 vtss_ewis_cons_act_set()

```
vtss_rc vtss_ewis_cons_act_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_ewis_cons_act_t *const cons_act )
```

Set consequent actions, i.e. how to handle AIS_L insertion and RDI_L backreporting.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>cons_act</i>	[IN] pointer to consequent actions.

Returns

Return code.

8.42.5.16 vtss_ewis_cons_act_get()

```
vtss_rc vtss_ewis_cons_act_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_ewis_cons_act_t *const cons_act )
```

Get the configured consequent actions.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>cons_act</i>	[OUT] pointer to consequent actions.

Returns

Return code.

8.42.5.17 vtss_ewis_section_txti_set()

```
vtss_rc vtss_ewis_section_txti_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_ewis_txti_t *const txti )
```

Set section transmitted Trail Trace Identifier. The transmitted trace identifier is aligned according to the specification, i.e 16 byte mode: In the first byte MSB = 1, and all other bytes MSB = 0. 64 byte mode: this is a text string terminated with CR/LF.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>txti</i>	[IN] pointer to transmitted tti.

Returns

Return code.

8.42.5.18 vtss_ewis_section_txi_get()

```
vtss_rc vtss_ewis_section_txi_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_ewis_tti_t *const txi )
```

Get the configured section transmitted Trail Trace Identifier.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>txti</i>	[OUT] pointer to transmitted tti.

Returns

Return code.

8.42.5.19 vtss_ewis_exp_sl_set()

```
vtss_rc vtss_ewis_exp_sl_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_ewis_sl_conf_t *const sl )
```

Set expected Signal label. The signal label is only configurable in SONET/SDH mode, in this mode the path overhead is not terminated, it is only monitored, i.e. only expected signal label is configurable.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>sl</i>	[IN] pointer to expected signal label.

Returns

Return code.

8.42.5.20 vtss_ewis_path_txi_set()

```
vtss_rc vtss_ewis_path_txi_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_ewis_tti_t *const txi )
```

Set Path Transmitted Trail Trace Identifier. The transmitted trace identifier is aligned according to the specification, i.e 16 byte mode: In the first byte MSB = 1, and all other bytes MSB = 0. 64 byte mode: this is a text string terminated with CR/LF. *

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>txi</i>	[IN] pointer to transmitted tti.

Returns

Return code.

8.42.5.21 vtss_ewis_path_txi_get()

```
vtss_rc vtss_ewis_path_txi_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_ewis_tti_t *const txi )
```

Get the configured Path Transmitted Trail Trace Identifier.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>txi</i>	[OUT] pointer to transmitted tti.

Returns

Return code.

8.42.5.22 vtss_ewis_test_mode_set()

```
vtss_rc vtss_ewis_test_mode_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_ewis_test_conf_t *const test_mode )
```

Set WIS test mode.

Note

This is useful for debugging purpose.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>test_mode</i>	[IN] Set WIS test mode (loopback and test patterns).

Returns

Return code.

Test pattern setup is applied to both TX (test generator) and RX (test analyzer)

8.42.5.23 vtss_ewis_test_mode_get()

```
vtss_rc vtss_ewis_test_mode_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_ewis_test_conf_t *const test_mode )
```

Get eWIS test mode.

Note

This is useful for debugging purpose.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>test_mode</i>	[OUT] Get eWIS test mode (loopback and test patterns).

Returns

Return code.

8.42.5.24 vtss_ewis_prbs31_err_inj_set()

```
vtss_rc vtss_ewis_prbs31_err_inj_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_ewis_prbs31_err_inj_t *const inj )
```

Inject eWIS PRBS31 errors.

Note

This is useful for debugging purpose.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>inj</i>	[IN] Defines the type of error injected.

Returns

Return code.

Test pattern setup is applied to both TX (test generator) and RX (test analyzer)

8.42.5.25 vtss_ewis_test_counter_get()

```
vtss_rc vtss_ewis_test_counter_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_ewis_test_status_t *const test_status )
```

Get eWIS test counter.

Note

This is useful for debugging purpose.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>test_status</i>	[OUT] Get eWIS test status (test pattern error counter, clear on read).

Returns

Return code.

Test pattern error counter is only used in prbs31 mode. In mixed frequency mode, the normal performance counters are maintained.

8.42.5.26 vtss_ewis_defects_get()

```
vtss_rc vtss_ewis_defects_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_ewis_defects_t *const def )
```

Get eWIS defects. Reports the current status of the defects.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>def</i>	[OUT] pointer to defect status structure.

Returns

Return code.

8.42.5.27 vtss_ewis_status_get()

```
vtss_rc vtss_ewis_status_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_ewis_status_t *const status )
```

Get eWIS fault and link status.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>status</i>	[OUT] pointer to status structure.

Returns

Return code.

8.42.5.28 vtss_ewis_section_acti_get()

```
vtss_rc vtss_ewis_section_acti_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_ewis_tti_t *const acti )
```

Get section received (accepted) Trail Trace Identifier. The received trace identifier is aligned according to the specification, i.e 16 byte mode: In the first byte MSB = 1, and all other bytes MSB = 0. (see G.707 section 9.2.2.2)
No CRC checksum verification is done in the API. 64 byte mode: this is a text string terminated with CR/LF.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>acti</i>	[OUT] pointer to accepted tti.

Returns

Return code.

8.42.5.29 vtss_ewis_path_acti_get()

```
vtss_rc vtss_ewis_path_acti_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_ewis_tti_t *const acti )
```

Get path received (accepted) Trail Trace Identifier.

The received trace identifier is aligned according to the specification, i.e 16 byte mode: In the first byte MSB = 1, and all other bytes MSB = 0. (see G.707 section 9.2.2.2) No CRC checksum verification is done in the API. 64 byte mode: this is a text string terminated with CR/LF.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>acti</i>	[OUT] pointer to accepted TTI.

Returns

Return code.

8.42.5.30 vtss_ewis_counter_get()

```
vtss_rc vtss_ewis_counter_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_ewis_counter_t *const counter )
```

Get free running eWIS counters.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>counter</i>	[OUT] pointer to counter structure.

Returns

Return code.

8.42.5.31 vtss_ewis_perf_get()

```
vtss_rc vtss_ewis_perf_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_ewis_perf_t *const perf )
```

Get eWIS counters per second (performance primitives). By default the source of PMTICK event (generation of 1 second) is configured to be internal. The values are accumulate for a period of 1 second and are then updated in the associated registers.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>perf</i>	[OUT] pointer to performance primitive structure.

Returns

Return code.

8.42.5.32 vtss_ewis_counter_threshold_set()

```
vtss_rc vtss_ewis_counter_threshold_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_ewis_counter_threshold_t *const threshold )
```

Set eWIS error counter thresholds per second. The PHY generates an interrupt once the error counter exceeds the configured threshold values. The units is frames per second. The threshold value configuration is possible for B1,B2,B3,REIP,REIL errors. Eg: a threshold value of 0 gives one interrupt for every 1 second if the error counter is atleast 1.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>threshold</i>	[IN] pointer to counter threshold structure.

Returns

Return code.

8.42.5.33 vtss_ewis_counter_threshold_get()

```
vtss_rc vtss_ewis_counter_threshold_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_ewis_counter_threshold_t *const threshold )
```

Get the configured eWIS error counter thresholds.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>threshold</i>	[OUT] pointer to eWIS error counters threshold structure.

Returns

Return code.

8.42.5.34 vtss_ewis_perf_mode_set()

```
vtss_rc vtss_ewis_perf_mode_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_ewis_perf_mode_t const * perf_mode )
```

Set the eWIS performance block counter modes.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>perf_mode</i>	[IN] Pointer to the modes of the all performance counters.

Returns

Return code.

8.42.5.35 vtss_ewis_perf_mode_get()

```
vtss_rc vtss_ewis_perf_mode_get (
    const vtss_inst_t inst,
```

```
const vtss_port_no_t port_no,
vtss_ewis_perf_mode_t *const perf_mode )
```

Get the eWIS performance block counter modes.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>perf_mode</i>	[OUT] Pointer to the modes of the all performance counters.

Returns

Return code.

8.43 vtss_api/include/vtss_xaui_api.h File Reference

XAUI API.

```
#include <vtss/api/types.h>
```

8.43.1 Detailed Description

XAUI API.

8.44 vtss_api/include/vtss_xfi_api.h File Reference

XFI API.

```
#include <vtss/api/types.h>
```

8.44.1 Detailed Description

XFI API.

Index

a_gpio
 vtss_phy_10g_auto_failover_conf_t, 220

AMPLITUDE_POINTS
 vtss_phy_10g_api.h, 808

ach_opt
 vtss_phy_ts_oam_engine_flow_conf_t, 384
 vtss_phy_ts_ptp_engine_flow_conf_t, 391

acknowledge
 vtss_phy_10g_clause_37_adv_t, 233
 vtss_port_clause_37_adv_t, 408

acl_hit
 vtss_packet_rx_info_t, 188

acl_idx
 vtss_packet_rx_info_t, 188

action
 vtss_ace_t, 53
 vtss_acl_port_conf_t, 60
 vtss_ece_t, 84
 vtss_phy_ts_engine_action_t, 349
 vtss_qce_t, 453
 vtss_vce_t, 515

action_gen
 vtss_phy_ts_engine_action_t, 349

action_ptp
 vtss_phy_ts_engine_action_t, 349

active
 vtss_irq_status_t, 154
 vtss_phy_10g_kr_status_aneg_t, 253
 vtss_phy_10g_prbs_mon_conf_t, 290

addr
 vtss_ip_addr_t, 147
 vtss_ipv6_t, 151
 vtss_mac_t, 166
 vtss_phy_ts_ip_conf_t, 374
 vtss_wol_mac_addr_t, 536

addr_match_mode
 vtss_phy_ts_eth_conf_t, 354

addr_match_select
 vtss_phy_ts_eth_conf_t, 354

address
 vtss_ip_network_t, 148
 vtss_ipv4_network_t, 148
 vtss_ipv6_network_t, 150

adv_10g
 vtss_phy_10g_base_kr_id_adv_abil_t, 224

adv_1g
 vtss_phy_10g_base_kr_id_adv_abil_t, 224

adv_dis
 port_custom_conf_t, 34

advertisement
 vtss_phy_10g_clause_37_control_t, 235
 vtss_port_clause_37_control_t, 410

agc
 vtss_phy_10g_ib_conf_t, 244

aged
 vtss_mac_table_entry_t, 167
 vtss_mac_table_status_t, 170

aggr_code
 vtss_packet_tx_info_t, 208

aggr_intrpt
 vtss_gpio_10g_gpio_mode_t, 140

aggr_rx_disable
 vtss_packet_frame_info_t, 177
 vtss_packet_port_info_t, 179

aggr_tx_disable
 vtss_packet_frame_info_t, 177
 vtss_packet_port_info_t, 180

ais_on_lof
 vtss_ewis_aisl_cons_act_s, 97

ais_on_los
 vtss_ewis_aisl_cons_act_s, 97

aisl
 vtss_ewis_cons_act_s, 102

amp_range
 vtss_phy_10g_vsphere_scan_status_t, 310

ampl
 vtss_phy_10g_base_kr_conf_t, 223

an_enable
 vtss_phy_10g_base_kr_autoneg_t, 221

an_reset
 vtss_phy_10g_base_kr_autoneg_t, 221

an_restart
 vtss_phy_10g_base_kr_autoneg_t, 221

ana_sync
 vtss_ewis_test_status_s, 128

aneg
 vtss_phy_10g_base_kr_status_t, 226
 vtss_phy_conf_t, 316
 vtss_port_status_t, 440

aneg_complete
 vtss_port_sgmii_aneg_t, 438
 vtss_port_status_t, 440

aneg_enable
 vtss_phy_tbi_conf_t, 342

aneg_pd_detect
 vtss_phy_media_serdes_pcs_ctrl_t, 331

aneg_restart
 vtss_phy_mac_serdes_pcs_ctrl_t, 328

apc_bit_mask
 vtss_phy_10g_ib_conf_t, 246

apc_eqz_offs_par_cfg
 vtss_phy_10g_mode_t, 268

apc_host_eqz_ld_ctrl
 vtss_phy_10g_mode_t, 268

apc_host_ld_ctrl
 vtss_phy_10g_mode_t, 267

apc_ib_regulator
 vtss_phy_10g_mode_t, 269

apc_ib_regulator_t
 vtss_phy_10g_api.h, 825

apc_line_eqz_ld_ctrl
 vtss_phy_10g_mode_t, 268

apc_line_ld_ctrl
 vtss_phy_10g_mode_t, 267

apc_offs_ctrl
 vtss_phy_10g_mode_t, 266

arp
 vtss_ace_frame_arp_t, 37
 vtss_ace_t, 54

arrived_tagged
 vtss_packet_rx_header_t, 183

asymmetric_pause
 vtss_phy_10g_clause_37_adv_t, 232
 vtss_phy_aneg_t, 312
 vtss_port_clause_37_adv_t, 408

auto_clear_ls
 vtss_phy_ts_init_conf_t, 370

automatic
 vtss_learn_mode_t, 164

autoneg
 port_custom_conf_t, 33
 vtss_phy_base_kr_train_aneg_t, 227
 vtss_phy_clause_37_status_t, 237

BOOLEAN_STORAGE_COUNT
 vtss_phy_10g_api.h, 808

BOOL
 types.h, 599

bad_crc
 vtss_phy_10g_pkt_mon_conf_t, 282

base_address
 vtss_l3_common_conf_t, 156

base_port_no
 vtss_phy_type_t, 398

ber
 vtss_phy_10g_pkt_mon_conf_t, 283
 vtss_phy_10g_vscope_scan_conf_t, 309

ber_cnt
 vtss_phy_pcs_cnt_t, 333

bin
 vtss_packet_tx_info_t, 210

bist_mode
 vtss_phy_10g_prbs_mon_conf_t, 289

bit_count
 vtss_sgpio_conf_t, 475

bit_errors
 vtss_phy_10g_ib_status_t, 247

block_lock
 vtss_phy_10g_kr_status_aneg_t, 255
 vtss_phy_10g_status_t, 304

block_lock_latched
 vtss_phy_pcs_cnt_t, 333

bmode
 vtss_sgpio_conf_t, 475

bottom_up
 vtss_phy_ts_mpls_conf_t, 378

bpdu_cpu_only
 vtss_packet_rx_reg_t, 203

bpdu_queue
 vtss_packet_rx_queue_map_t, 200

bpdu_reg
 vtss_packet_rx_port_conf_t, 198

bridge
 vtss_port_counters_t, 415

broadcast
 vtss_policer_ext_t, 403

bypass_in_api
 vtss_phy_10g_fifo_sync_t, 239

bytes
 vtss_counter_pair_t, 67

c
 vtss_phy_10g_ib_conf_t, 244

c0
 vtss_phy_10g_base_kr_conf_t, 222

c0_ob_tap_result
 vtss_phy_10g_kr_status_train_t, 257

c1
 vtss_phy_10g_base_kr_conf_t, 223

c_ctrl
 vtss_phy_10g_ob_status_t, 274

c_intrpt
 vtss_gpio_10g_gpio_mode_t, 140

CALIB_DONE
 vtss_phy_ts_nphase_status_t, 381

CALIB_ERR
 vtss_phy_ts_nphase_status_t, 380

CHIP_PORT_UNUSED
 vtss_port_api.h, 1034

cal_done
 vtss_lcpll_status_t, 163

cal_error
 vtss_lcpll_status_t, 163
 vtss_rcpll_status_t, 466

cal_not_done
 vtss_rcpll_status_t, 466

cb
 vtss_ts_timestamp_alloc_t, 490

cbs
 vtss_dlb_policer_conf_t, 72

cf
 vtss_dlb_policer_conf_t, 72

cf_update
 vtss_phy_ts_ptp_engine_action_t, 389

cfg
 vtss_phy_conf_1g_t, 315

cfg0
 vtss_phy_10g_mode_t, 267

cfi
 vtss_ace_vlan_t, 56
 vtss_tci_t, 482

channel_conf
 vtss_phy_10g_init_parm_t, 251

channel_id
 vtss_phy_10g_auto_failover_conf_t, 219
 vtss_phy_10g_id_t, 250
 vtss_phy_10g_mode_t, 264
 vtss_phy_type_t, 398

channel_map
 vtss_phy_ts_engine_flow_conf_t, 350
 vtss_phy_ts_generic_action_t, 364
 vtss_phy_ts_oam_engine_action_t, 382
 vtss_phy_ts_ptp_engine_action_t, 388

channel_type
 vtss_phy_ts_ach_conf_t, 345

chip_no
 vtss_debug_info_t, 68
 vtss_debug_lock_t, 70
 vtss_packet_rx_meta_t, 194
 vtss_port_map_t, 427

chip_port
 vtss_port_map_t, 427
 vtss_vstax_tx_header_t, 535

chk_ing_modified
 vtss_phy_ts_init_conf_t, 371

cir
 vtss_dlb_policer_conf_t, 72

ckout_sel
 vtss_phy_10g_ckout_conf_t, 231

ckout_sel_
 vtss_phy_10g_api.h, 832

ckout_sel_t
 vtss_phy_10g_api.h, 821

clear
 vtss_debug_info_t, 69

clk_freq
 vtss_phy_ts_init_conf_t, 368

clk_mode
 vtss_phy_ts_ptp_engine_action_t, 389

clk_mstr_t
 vtss_phy_10g_api.h, 826

clk_sel_no
 vtss_phy_10g_host_clk_conf_t, 242
 vtss_phy_10g_line_clk_conf_t, 260

clk_src
 vtss_phy_ts_init_conf_t, 368

cm
 vtss_dlb_policer_conf_t, 71

cm1
 vtss_phy_10g_base_kr_conf_t, 222

cm_ob_tap_result
 vtss_phy_10g_kr_status_train_t, 257

cmeff_disable
 vtss_vstax_conf_t, 527

comm_opt
 vtss_phy_ts_ach_conf_t, 345
 vtss_phy_ts_eth_conf_t, 353
 vtss_phy_ts_gen_conf_t, 362
 vtss_phy_ts_ip_conf_t, 373
 vtss_phy_ts_mpls_conf_t, 376

complete
 vtss_phy_10g_clause_37_status_t, 236
 vtss_phy_10g_kr_status_aneg_t, 253
 vtss_phy_10g_kr_status_train_t, 257

config_bit_mask
 vtss_phy_10g_ib_conf_t, 246

connector_enable
 vtss_phy_loopback_t, 324

context
 vtss_ts_timestamp_alloc_t, 490
 vtss_ts_timestamp_t, 491

copper
 vtss_port_status_t, 441

copy_to_cpu
 vtss_mac_table_entry_t, 167

corrected_block_cnt
 vtss_phy_10g_kr_status_fec_t, 256

cos
 vtss_packet_rx_info_t, 188
 vtss_packet_tx_info_t, 209

cp_ob_tap_result
 vtss_phy_10g_kr_status_train_t, 257

cpu
 vtss_acl_action_t, 57
 vtss_learn_mode_t, 165

cpu_once
 vtss_acl_action_t, 57

cpu_queue
 vtss_acl_action_t, 57
 vtss_mac_table_entry_t, 167
 vtss_policer_ext_t, 405

cs_wait_ns
 vtss_pi_conf_t, 402

cu_bad
 vtss_phy_statistic_t, 339

cu_good
 vtss_phy_statistic_t, 339

cur_version
 vtss_restart_status_t, 469

cw_en
 vtss_phy_ts_mpls_conf_t, 376

d_filter
 vtss_phy_10g_mode_t, 267

d_fltr
 vtss_phy_10g_ob_status_t, 275

dais_l
 vtss_ewis_defects_s, 107

dais_p
 vtss_ewis_defects_s, 107

data
 vtss_ace_frame_etype_t, 40
 vtss_ace_frame_ipv4_t, 42

vtss_ace_frame_ipv6_t, 46
 vtss_phy_ts_gen_conf_t, 363
 vtss_phy_ts_generic_action_t, 364
 vtss_qce_frame_etype_t, 444
 vtss_qce_frame_llc_t, 448
 vtss_qce_frame_snap_t, 448
 vtss_vce_frame_etype_t, 506
 vtss_vce_frame_llc_t, 510
 vtss_vce_frame_snap_t, 510
 ddr_mode
 vtss_phy_10g_mode_t, 270
 ddr_mode_t
 vtss_phy_10g_api.h, 825
 default_dei
 vtss_qos_port_conf_t, 460
 default_dpl
 vtss_qos_port_conf_t, 460
 default_prio
 vtss_qos_port_conf_t, 460
 dei
 vtss_ece_inner_tag_t, 78
 vtss_ece_outer_tag_t, 82
 vtss_ece_tag_t, 85
 vtss_mirror_conf_t, 172
 vtss_qce_tag_t, 455
 vtss_vce_tag_t, 516
 vtss_vlan_tag_t, 522
 dei_colouring
 vtss_evc_port_conf_t, 96
 delaym_type
 vtss_phy_ts_ietf_mpls_ach_oam_conf_t, 367
 vtss_phy_ts_ptp_engine_action_t, 389
 vtss_phy_ts_y1731_oam_conf_t, 395
 des_interface_width
 vtss_phy_10g_prbs_mon_conf_t, 288
 dest_ip
 vtss_phy_ts_fifo_sig_t, 361
 dest_mac
 vtss_phy_ts_fifo_sig_t, 361
 destination
 vtss_ipv4_uc_t, 150
 vtss_ipv6_uc_t, 152
 vtss_irq_conf_t, 153
 vtss_mac_table_entry_t, 167
 detect_only
 vtss_phy_ts_fifo_conf_t, 358
 device_feature_status
 vtss_phy_10g_id_t, 250
 dfais_p
 vtss_ewis_defects_s, 108
 dfe1
 vtss_phy_10g_ib_conf_t, 244
 dfe2
 vtss_phy_10g_ib_conf_t, 244
 dfe3
 vtss_phy_10g_ib_conf_t, 245
 dfe4
 vtss_phy_10g_ib_conf_t, 245
 dfplm_p
 vtss_ewis_defects_s, 108
 dfuneq_p
 vtss_ewis_defects_s, 109
 dgroup_no
 vtss_dgroup_port_conf_t, 70
 dig_offset_reg
 vtss_phy_10g_mode_t, 266
 dip
 vtss_ace_frame_arp_t, 39
 vtss_ace_frame_ipv4_t, 42
 vtss_l3_neighbour_t, 159
 dir
 vtss_ece_action_t, 73
 disable
 vtss_phy_mac_serdes_pcs_cntl_t, 327
 discard
 vtss_learn_mode_t, 165
 divider
 vtss_sync_clock_out_t, 481
 dlcd_p
 vtss_ewis_defects_s, 108
 dlof_s
 vtss_ewis_defects_s, 106
 dlop_p
 vtss_ewis_defects_s, 107
 dlos_s
 vtss_ewis_defects_s, 106
 dma_enable
 vtss_packet_dma_conf_t, 175
 dmac
 vtss_ace_frame_etype_t, 39
 vtss_ace_frame_llc_t, 49
 vtss_ace_frame_snap_t, 51
 vtss_ece_mac_t, 81
 vtss_l3_neighbour_t, 159
 vtss_phy_10g_pkt_gen_conf_t, 280
 dmac_bc
 vtss_ace_t, 53
 vtss_qce_mac_t, 452
 vtss_vce_mac_t, 514
 dmac_dip
 vtss_vcl_port_conf_t, 517
 dmac_enable
 vtss_aggr_mode_t, 61
 dmac_match
 vtss_ace_frame_arp_t, 38
 dmac_mc
 vtss_ace_t, 53
 vtss_qce_mac_t, 452
 vtss_vce_mac_t, 514
 domain
 vtss_phy_ts_ptp_conf_t, 387
 domain_num
 vtss_phy_ts_fifo_sig_t, 360
 doof_s
 vtss_ewis_defects_s, 106
 dot1dTpPortInDiscards

vtss_port_bridge_counters_t, 407
dot3ControllnUnknownOpcodes
 vtss_port_ethernet_like_counters_t, 417
dot3InPauseFrames
 vtss_port_ethernet_like_counters_t, 417
dot3OutPauseFrames
 vtss_port_ethernet_like_counters_t, 419
dot3StatsAlignmentErrors
 vtss_port_ethernet_like_counters_t, 416
dot3StatsCarrierSenseErrors
 vtss_port_ethernet_like_counters_t, 419
dot3StatsDeferredTransmissions
 vtss_port_ethernet_like_counters_t, 418
dot3StatsExcessiveCollisions
 vtss_port_ethernet_like_counters_t, 418
dot3StatsFCSErrors
 vtss_port_ethernet_like_counters_t, 417
dot3StatsFrameTooLongs
 vtss_port_ethernet_like_counters_t, 417
dot3StatsLateCollisions
 vtss_port_ethernet_like_counters_t, 418
dot3StatsMultipleCollisionFrames
 vtss_port_ethernet_like_counters_t, 418
dot3StatsSingleCollisionFrames
 vtss_port_ethernet_like_counters_t, 418
dot3StatsSymbolErrors
 vtss_port_ethernet_like_counters_t, 417
dp
 vtss_packet_tx_info_t, 214
 vtss_qce_action_t, 442
 vtss_vstax_tx_header_t, 535
dp_bypass_level
 vtss_policer_ext_t, 403
dp_enable
 vtss_qce_action_t, 442
dp_level_map
 vtss_qos_port_conf_t, 461
dplm_p
 vtss_ewis_defects_s, 108
dport
 vtss_ace_frame_ipv4_t, 43
 vtss_ace_frame_ipv6_t, 47
 vtss_qce_frame_ipv4_t, 445
 vtss_qce_frame_ipv6_t, 447
 vtss_vce_frame_ipv4_t, 507
 vtss_vce_frame_ipv6_t, 509
dport_mask
 vtss_phy_ts_ip_conf_t, 373
dport_val
 vtss_phy_ts_ip_conf_t, 373
drdi_l
 vtss_ewis_defects_s, 107
drdi_p
 vtss_ewis_defects_s, 108
ds
 vtss_ace_frame_ipv4_t, 42
 vtss_ace_frame_ipv6_t, 46
 vtss_phy_ts_ietf_mpls_ach_oam_conf_t, 367
dscp
 vtss_ece_frame_ipv4_t, 75
 vtss_ece_frame_ipv6_t, 76
 vtss_qce_action_t, 443
 vtss_qce_frame_ipv4_t, 445
 vtss_qce_frame_ipv6_t, 446
 vtss_vce_frame_ipv4_t, 507
 vtss_vce_frame_ipv6_t, 508
dscp_class_enable
 vtss_qos_port_conf_t, 461
dscp_dp_level_map
 vtss_qos_conf_t, 456
dscp_emode
 vtss_qos_port_conf_t, 461
dscp_enable
 vtss_qce_action_t, 443
dscp_mode
 vtss_qos_port_conf_t, 461
dscp_qos_class_map
 vtss_qos_conf_t, 456
dscp_qos_map
 vtss_qos_conf_t, 456
dscp_remap
 vtss_qos_conf_t, 457
dscp_remark
 vtss_qos_conf_t, 457
dscp_translate
 vtss_qos_port_conf_t, 462
dscp_translate_map
 vtss_qos_conf_t, 457
dscp_trust
 vtss_qos_conf_t, 456
dst_port_mask
 vtss_packet_tx_info_t, 206
dual_media_fiber_speed
 port_custom_conf_t, 34
duneq_p
 vtss_ewis_defects_s, 107
dwrr_enable
 vtss_qos_port_conf_t, 463
ebs
 vtss_dlb_policer_conf_t, 73
edc_fw_api_load
 vtss_phy_10g_fw_status_t, 241
edc_fw_chksum
 vtss_phy_10g_fw_status_t, 240
edc_fw_load
 vtss_phy_10g_mode_t, 265
edc_fw_rev
 vtss_phy_10g_fw_status_t, 240
eee_ena
 vtss_eee_port_conf_t, 86
eee_ena_phy
 vtss_phy_eee_conf_t, 320
eee_fast_queues
 vtss_eee_port_conf_t, 86
eee_mode
 vtss_phy_eee_conf_t, 319

egr_fcs_err
 vtss_phy_ts_stats_t, 393
 egr_frm_mod_cnt
 vtss_phy_ts_stats_t, 394
 egr_pream_shrink_err
 vtss_phy_ts_stats_t, 393
 eir
 vtss_dlb_policer_conf_t, 72
 en_ob
 vtss_phy_10g_base_kr_conf_t, 223
 ena_ifh_header
 vtss_port_ifh_t, 426
 enable
 port_custom_conf_t, 33
 vtss_dlb_policer_conf_t, 71
 vtss_ece_outer_tag_t, 82
 vtss_mac_table_entry_t, 168
 vtss_npi_conf_t, 173
 vtss_packet_rx_queue_npi_conf_t, 202
 vtss_phy_10g_auto_failover_conf_t, 220
 vtss_phy_10g_base_kr_training_t, 228
 vtss_phy_10g_ckout_conf_t, 231
 vtss_phy_10g_clause_37_control_t, 235
 vtss_phy_10g_kr_status_fec_t, 256
 vtss_phy_10g_lane_sync_conf_t, 258
 vtss_phy_10g_loopback_t, 261
 vtss_phy_10g_pkt_gen_conf_t, 278
 vtss_phy_10g_pkt_mon_conf_t, 282
 vtss_phy_10g_prbs_gen_conf_t, 285
 vtss_phy_10g_prbs_mon_conf_t, 287
 vtss_phy_10g_sckout_conf_t, 293
 vtss_phy_10g_srefclk_mode_t, 301
 vtss_phy_10g_vscope_conf_t, 306
 vtss_phy_ltc_freq_synth_s, 326
 vtss_phy_ts_generic_action_t, 364
 vtss_phy_ts_nphase_status_t, 380
 vtss_phy_ts_oam_engine_action_t, 382
 vtss_phy_ts_ptp_engine_action_t, 388
 vtss_port_clause_37_control_t, 409
 vtss_red_t, 467
 vtss_sync_clock_in_t, 480
 vtss_sync_clock_out_t, 481
 vtss_ts_ext_clock_mode_t, 486
 enable_pass_thru
 vtss_phy_10g_clause_37_control_t, 235
 vtss_phy_10g_mode_t, 273
 enabled
 vtss_sgpi_port_conf_t, 476
 encaps_type
 vtss_phy_ts_eng_init_conf_t, 347
 end
 vtss_phy_ts_mpls_conf_t, 378
 eng_minE
 vtss_phy_ts_fifo_conf_t, 359
 eng_mode
 vtss_phy_ts_engine_flow_conf_t, 350
 eng_recov
 vtss_phy_ts_fifo_conf_t, 358
 eng_used
 vtss_phy_ts_eng_init_conf_t, 347
 err_blk_cnt
 vtss_phy_pcs_cnt_t, 333
 error_counter
 vtss_phy_10g_pcs_prbs_mon_conf_t, 277
 error_free_x
 vtss_phy_10g_vsphere_scan_status_t, 310
 error_free_y
 vtss_phy_10g_vsphere_scan_status_t, 310
 error_states
 vtss_phy_10g_prbs_mon_conf_t, 288
 error_status
 vtss_phy_10g_prbs_mon_conf_t, 289
 error_thres
 vtss_phy_10g_vsphere_conf_t, 306
 errors
 vtss_phy_10g_vsphere_scan_status_t, 310
 eth1_opt
 vtss_phy_ts_generic_flow_conf_t, 366
 vtss_phy_ts_oam_engine_flow_conf_t, 383
 vtss_phy_ts_ptp_engine_flow_conf_t, 390
 eth2_opt
 vtss_phy_ts_oam_engine_flow_conf_t, 384
 vtss_phy_ts_ptp_engine_flow_conf_t, 390
 ethernet
 vtss_ace_frame_arp_t, 38
 ethernet_like
 vtss_port_counters_t, 415
 etype
 vtss_ace_frame_etype_t, 40
 vtss_ace_t, 54
 vtss_packet_rx_meta_t, 195
 vtss_phy_10g_pkt_gen_conf_t, 279
 vtss_phy_ts_eth_conf_t, 353
 vtss_qce_frame_etype_t, 444
 vtss_qce_key_t, 450
 vtss_vce_frame_etype_t, 505
 vtss_vce_key_t, 512
 evc_id
 vtss_ece_action_t, 74
 evnt
 vtss_phy_10g_auto_failover_conf_t, 219
 ewis_cnt_cfg
 vtss_ewis_static_conf_s, 125
 ewis_cntr_thresh_conf
 vtss_ewis_conf_s, 101
 ewis_init_done
 vtss_ewis_conf_s, 98
 ewis_lof_ctrl1
 vtss_ewis_static_conf_s, 124
 ewis_lof_ctrl2
 vtss_ewis_static_conf_s, 124
 ewis_mode
 vtss_ewis_conf_s, 99
 ewis_mode_ctrl
 vtss_ewis_static_conf_s, 123
 ewis_pmtick_ctrl

vtss_ewis_static_conf_s, 125
ewis_rx_ctrl1
 vtss_ewis_static_conf_s, 122
ewis_rx_err_frc1
 vtss_ewis_static_conf_s, 124
ewis_rx_frm_ctrl1
 vtss_ewis_static_conf_s, 124
ewis_rx_frm_ctrl2
 vtss_ewis_static_conf_s, 124
ewis_tx_a1_a2
 vtss_ewis_static_conf_s, 123
ewis_tx_c2_h1
 vtss_ewis_static_conf_s, 123
ewis_tx_h2_h3
 vtss_ewis_static_conf_s, 123
ewis_tx_z0_e1
 vtss_ewis_static_conf_s, 123
ewis_txctrl1
 vtss_ewis_static_conf_s, 122
ewis_txctrl2
 vtss_ewis_static_conf_s, 122
exc_col_cont
 port_custom_conf_t, 34
 vtss_port_conf_t, 413
excess_enable
 vtss_qos_port_conf_t, 459
exp_sl
 vtss_ewis_conf_s, 100
exsl
 vtss_ewis_sl_conf_s, 121
external
 vtss_irq_conf_t, 153

f_ebc_thr_l
 vtss_ewis_counter_threshold_s, 105
f_ebc_thr_p
 vtss_ewis_counter_threshold_s, 105
FALSE
 types.h, 580
family
 vtss_phy_10g_id_t, 250
fan_low_pol
 vtss_fan_conf_t, 137
fan_open_col
 vtss_fan_conf_t, 138
fan_pwm_freq
 vtss_fan_conf_t, 137
far_end_enable
 vtss_phy_loopback_t, 324
fast_link_stat_ena
 vtss_phy_mac_serdes_pcs_cntl_t, 329
fault
 vtss_ewis_cons_act_s, 102
 vtss_ewis_status_s, 126
fault_on_aisl
 vtss_ewis_fault_cons_act_s, 111
fault_on_aisp
 vtss_ewis_fault_cons_act_s, 111
fault_on_feaisp
 vtss_ewis_fault_cons_act_s, 110
fault_on_feplmp
 vtss_ewis_fault_cons_act_s, 110
fault_on_lcdp
 vtss_ewis_fault_cons_act_s, 111
fault_on_lof
 vtss_ewis_fault_cons_act_s, 110
fault_on_lopp
 vtss_ewis_fault_cons_act_s, 111
fault_on_los
 vtss_ewis_fault_cons_act_s, 110
fault_on_plmp
 vtss_ewis_fault_cons_act_s, 111
fault_on_rdil
 vtss_ewis_fault_cons_act_s, 110
fault_on_sef
 vtss_ewis_fault_cons_act_s, 110
fcs
 vtss_packet_rx_meta_t, 195
fdx
 port_custom_conf_t, 33
 vtss_phy_10g_clause_37_adv_t, 232
 vtss_phy_forced_t, 322
 vtss_port_clause_37_adv_t, 408
 vtss_port_conf_t, 412
 vtss_port_sgmii_aneg_t, 437
 vtss_port_status_t, 440
fdx_gap
 vtss_port_frame_gaps_t, 421
fec
 vtss_phy_10g_base_kr_status_t, 226
fec_abil
 vtss_phy_10g_base_kr_id_adv_abil_t, 225
fec_enable
 vtss_phy_10g_kr_status_aneg_t, 254
fec_req
 vtss_phy_10g_base_kr_id_adv_abil_t, 225
fiber
 vtss_port_status_t, 441
fid
 vtss_vlan_vid_conf_t, 526
file
 vtss_api_lock_t, 64
fill_level
 vtss_eee_port_counter_t, 88
fill_level_get
 vtss_eee_port_counter_t, 88
fill_level_thres
 vtss_eee_port_counter_t, 88
filter
 vtss_packet_port_filter_t, 178
 vtss_phy_10g_auto_failover_conf_t, 220
flf
 vtss_phy_conf_t, 317
flooded
 vtss_policer_ext_t, 404
flow_conf
 vtss_phy_ts_engine_flow_conf_t, 351

flow_control
 port_custom_conf_t, 33
 vtss_policer_ext_t, 405
 vtss_port_conf_t, 412
flow_en
 vtss_phy_ts_eth_conf_t, 353
 vtss_phy_ts_gen_conf_t, 362
 vtss_phy_ts_ip_conf_t, 373
 vtss_phy_ts_mpls_conf_t, 376
flow_end_index
 vtss_phy_ts_eng_init_conf_t, 348
flow_id
 vtss_phy_ts_generic_action_t, 364
flow_match_mode
 vtss_phy_ts_eng_init_conf_t, 347
flow_offset
 vtss_phy_ts_gen_conf_t, 362
flow_opt
 vtss_phy_ts_eth_conf_t, 357
 vtss_phy_ts_gen_conf_t, 363
 vtss_phy_ts_ip_conf_t, 375
 vtss_phy_ts_mpls_conf_t, 379
flow_st_index
 vtss_phy_ts_eng_init_conf_t, 347
fltr_val
 vtss_phy_10g_auto_failover_conf_t, 220
force
 vtss_phy_reset_conf_t, 336
force_adv_ability
 vtss_phy_mac_serdes_pcs_ctrl_t, 328
 vtss_phy_media_serdes_pcs_ctrl_t, 331
force_ais
 vtss_ewis_line_force_mode_s, 113
 vtss_ewis_line_tx_force_mode_s, 114
force_ams_sel
 vtss_phy_conf_t, 318
force_fefi
 vtss_phy_media_serdes_pcs_ctrl_t, 332
force_fefi_value
 vtss_phy_media_serdes_pcs_ctrl_t, 332
force_hls
 vtss_phy_media_serdes_pcs_ctrl_t, 332
force_mode
 vtss_ewis_conf_s, 99
force_rdi
 vtss_ewis_line_force_mode_s, 114
 vtss_ewis_line_tx_force_mode_s, 115
 vtss_ewis_path_force_mode_s, 116
force_uneq
 vtss_ewis_path_force_mode_s, 115
forced
 vtss_phy_conf_t, 316
forward
 vtss_acl_action_t, 58
frag
 vtss_phy_10g_pkt_mon_conf_t, 283
fragment
 vtss_ace_frame_ipv4_t, 41
 vtss_qce_frame_ipv4_t, 445
 vtss_vce_frame_ipv4_t, 506
frame
 vtss_ace_t, 55
 vtss_ece_key_t, 80
 vtss_qce_key_t, 451
 vtss_vce_key_t, 513
frame_gaps
 vtss_port_conf_t, 411
frame_length_chk
 port_custom_conf_t, 35
 vtss_port_conf_t, 413
frame_rate
 vtss_policer_ext_t, 403
frame_single
 vtss_phy_10g_pkt_gen_conf_t, 279
frame_type
 vtss_vlan_port_conf_t, 521
frames
 vtss_counter_pair_t, 67
 vtss_phy_10g_pkt_gen_conf_t, 279
freeze
 vtss_phy_10g_apc_status_t, 218
freeze_bit_mask
 vtss_phy_10g_ib_conf_t, 246
freq
 vtss_phy_10g_ckout_conf_t, 230
 vtss_phy_10g_sckout_conf_t, 293
 vtss_phy_10g_srefclk_mode_t, 301
 vtss_phy_clock_conf_t, 314
 vtss_ts_ext_clock_mode_t, 487
frm_len
 vtss_packet_tx_info_t, 207
frst_lvl_after_top
 vtss_phy_ts_mpls_conf_t, 377
frst_lvl_before_end
 vtss_phy_ts_mpls_conf_t, 378
fsm_lock
 vtss_lcpll_status_t, 163
fsm_stat
 vtss_lcpll_status_t, 163
full
 vtss_debug_info_t, 68
function
 vtss_api_lock_t, 64
fwd_enable
 vtss_mirror_conf_t, 171
fwd_mode
 vtss_vstax_tx_header_t, 534
gain
 vtss_phy_10g_ib_conf_t, 243
gain_stat
 vtss_lcpll_status_t, 164
gainadj
 vtss_phy_10g_ib_conf_t, 243
garp_cpu_only
 vtss_packet_rx_reg_t, 203
garp_queue

vtss_packet_rx_queue_map_t, 200
garp_reg
 vtss_packet_rx_port_conf_t, 198
gen
 vtss_phy_ts_engine_flow_conf_t, 351
gen_conf
 vtss_phy_ts_engine_action_t, 349
gen_opt
 vtss_phy_ts_generic_flow_conf_t, 366
generate
 vtss_port_flow_control_conf_t, 420
generate_pause
 vtss_aneg_t, 63
glag_no
 vtss_packet_frame_info_t, 176
 vtss_packet_port_info_t, 179
 vtss_packet_rx_info_t, 185
 vtss_vstax_rx_header_t, 532
 vtss_vstax_tx_header_t, 535
good_crc
 vtss_phy_10g_pkt_mon_conf_t, 282
group
 vtss_debug_info_t, 68
group_id
 vtss_vlan_trans_grp2vlan_conf_t, 523
 vtss_vlan_trans_port2grp_conf_t, 524
grp_map
 vtss_packet_rx_conf_t, 181

h_apc_conf
 vtss_phy_10g_mode_t, 272
h_clk_src
 vtss_phy_10g_mode_t, 271
h_ib_conf
 vtss_phy_10g_mode_t, 272
h_media
 vtss_phy_10g_mode_t, 272
h_offset_guard
 vtss_phy_10g_mode_t, 269
h_pcsc
 vtss_phy_10g_serdes_status_t, 300
h_pll5g_fsm_lock
 vtss_phy_10g_serdes_status_t, 295
h_pll5g_fsm_stat
 vtss_phy_10g_serdes_status_t, 295
h_pll5g_gain
 vtss_phy_10g_serdes_status_t, 295
h_pll5g_lock_status
 vtss_phy_10g_serdes_status_t, 295
h_pma
 vtss_phy_10g_serdes_status_t, 299
h_rx_rcpll_fsm_status
 vtss_phy_10g_serdes_status_t, 297
h_rx_rcpll_lock_status
 vtss_phy_10g_serdes_status_t, 296
h_rx_rcpll_range
 vtss_phy_10g_serdes_status_t, 296
h_rx_rcpll_vco_load
 vtss_phy_10g_serdes_status_t, 297

h_tx_rcpll_fsm_status
 vtss_phy_10g_serdes_status_t, 298
h_tx_rcpll_lock_status
 vtss_phy_10g_serdes_status_t, 298
h_tx_rcpll_range
 vtss_phy_10g_serdes_status_t, 298
h_tx_rcpll_vco_load
 vtss_phy_10g_serdes_status_t, 298
hdx
 vtss_phy_10g_clause_37_adv_t, 232
 vtss_port_clause_37_adv_t, 408
 vtss_port_sgmii_aneg_t, 437
hdx_gap_1
 vtss_port_frame_gaps_t, 421
hdx_gap_2
 vtss_port_frame_gaps_t, 421
high
 vtss_phy_timestamp_t, 342
 vtss_vcap_udp_tcp_t, 501
 vtss_vcap_vr_t, 503
high_ber_latched
 vtss_phy_pcs_cnt_t, 333
high_duty_cycle
 vtss_phy_ltc_freq_synth_s, 326
high_input_gain
 vtss_phy_10g_mode_t, 264
hints
 vtss_packet_rx_info_t, 184
hl_clk_synth
 vtss_phy_10g_mode_t, 264
host
 vtss_phy_10g_clause_37_cmn_status_t, 234
 vtss_phy_10g_clause_37_control_t, 235
host_kr
 vtss_phy_10g_base_kr_train_aneg_t, 227
host_rx
 vtss_phy_10g_polarity_inv_t, 284
host_tx
 vtss_phy_10g_polarity_inv_t, 284
hpcsc
 vtss_phy_10g_status_t, 303
hpcsc_1g
 vtss_phy_10g_status_t, 303
hpma
 vtss_phy_10g_status_t, 302
hw_cnt
 vtss_os_timestamp_t, 174
hw_tstamp
 vtss_packet_rx_info_t, 190
hw_tstamp_decoded
 vtss_packet_rx_info_t, 190

i16
 types.h, 598
i32
 types.h, 598
i64
 types.h, 598
i8

types.h, 597
i_cpu_en
 vtss_phy_reset_conf_t, 337
i_tag
 vtss_phy_ts_eth_conf_t, 357
ib_conf
 vtss_phy_10g_ib_status_t, 247
 vtss_phy_10g_mode_t, 266
ib_cterm_ena
 vtss_serd़es_macro_conf_t, 473
ib_ini_lp
 vtss_phy_10g_mode_t, 267
ib_max_lp
 vtss_phy_10g_mode_t, 268
ib_min_lp
 vtss_phy_10g_mode_t, 268
ib_par_cfg, 31
 max, 32
 min, 31
 value, 31
ib_storage
 vtss_phy_10g_ib_storage_t, 248
ib_storage_bool
 vtss_phy_10g_ib_storage_t, 248
icpu_activity
 vtss_phy_10g_fw_status_t, 241
id
 vtss_ace_t, 52
 vtss_ece_t, 83
 vtss_qce_t, 453
 vtss_ts_timestamp_t, 491
 vtss_vce_t, 515
ietf_oam_conf
 vtss_phy_ts_oam_engine_action_t, 382
if_group
 vtss_port_counters_t, 415
if_type
 vtss_port_conf_t, 411
ifInBroadcastPkts
 vtss_port_if_group_counters_t, 423
ifInDiscards
 vtss_port_if_group_counters_t, 423
ifInErrors
 vtss_port_if_group_counters_t, 424
ifInMulticastPkts
 vtss_port_if_group_counters_t, 423
ifInNUcastPkts
 vtss_port_if_group_counters_t, 423
ifInOctets
 vtss_port_if_group_counters_t, 422
ifInUcastPkts
 vtss_port_if_group_counters_t, 423
ifOutBroadcastPkts
 vtss_port_if_group_counters_t, 424
ifOutDiscards
 vtss_port_if_group_counters_t, 425
ifOutErrors
 vtss_port_if_group_counters_t, 425
ifOutMulticastPkts
 vtss_port_if_group_counters_t, 424
ifOutNUcastPkts
 vtss_port_if_group_counters_t, 425
ifOutOctets
 vtss_port_if_group_counters_t, 424
ifOutUcastPkts
 vtss_port_if_group_counters_t, 424
ifh
 vtss_packet_tx_ifh_t, 205
igmp_cpu_only
 vtss_packet_rx_reg_t, 204
igmp_queue
 vtss_packet_rx_queue_map_t, 200
in_range
 vtss_vcap_udp_tcp_t, 500
in_sig
 vtss_gpio_10g_gpio_mode_t, 140
incomplete
 vtss_phy_10g_prbs_mon_conf_t, 290
incr_levn
 vtss_phy_10g_jitter_conf_t, 252
ingr_fcs_err
 vtss_phy_ts_stats_t, 393
ingr_frm_mod_cnt
 vtss_phy_ts_stats_t, 394
ingr_pream_shrink_err
 vtss_phy_ts_stats_t, 393
ingress
 vtss_phy_10g_pkt_gen_conf_t, 279
ingress_filter
 vtss_vlan_port_conf_t, 521
inhibit_odd_start
 vtss_phy_mac_serд_pcs_cntl_t, 330
 vtss_phy_media_serд_pcs_cntl_t, 331
inner_tag
 vtss_ece_action_t, 74
 vtss_ece_key_t, 79
 vtss_phy_ts_eth_conf_t, 357
inner_tag_type
 vtss_phy_ts_eth_conf_t, 355
input
 vtss_gpio_10g_gpio_mode_t, 139
inst
 vtss_api_lock_t, 64
instable
 vtss_phy_10g_prbs_mon_conf_t, 290
int_fmt
 vtss_ts_internal_mode_t, 488
int_pol_high
 vtss_sgpio_port_conf_t, 476
interface
 vtss_phy_10g_mode_t, 263
ip
 vtss_ace_frame_arp_t, 38
ip1_opt
 vtss_phy_ts_ptp_engine_flow_conf_t, 390
ip2_opt

vtss_phy_ts_ptp_engine_flow_conf_t, 390
ip_addr
 vtss_phy_ts_ip_conf_t, 375
ip_enable
 vtss_phy_ts_sertod_conf_t, 392
ip_mode
 vtss_phy_ts_ip_conf_t, 372
ipg_len
 vtss_phy_10g_pkt_gen_conf_t, 280
ipmc_ctrl_cpu_copy
 vtss_packet_rx_reg_t, 203
ipmc_ctrl_queue
 vtss_packet_rx_queue_map_t, 200
ipv4
 vtss_ace_t, 54
 vtss_ece_key_t, 80
 vtss_ip_addr_t, 146
 vtss_phy_ts_ip_conf_t, 374
 vtss_qce_key_t, 451
 vtss_vce_key_t, 512
ipv4_icmp_redirect_enable
 vtss_l3_rleg_conf_t, 160
ipv4_uc
 vtss_routing_entry_t, 470
ipv4_unicast_enable
 vtss_l3_rleg_conf_t, 160
ipv4uc_received_frames
 vtss_l3_counters_t, 157
ipv4uc_received_octets
 vtss_l3_counters_t, 157
ipv4uc_transmitted_frames
 vtss_l3_counters_t, 158
ipv4uc_transmitted_octets
 vtss_l3_counters_t, 157
ipv6
 vtss_ace_t, 55
 vtss_ece_key_t, 80
 vtss_ip_addr_t, 147
 vtss_phy_ts_ip_conf_t, 374
 vtss_qce_key_t, 451
 vtss_vce_key_t, 513
ipv6_icmp_redirect_enable
 vtss_l3_rleg_conf_t, 161
ipv6_uc
 vtss_routing_entry_t, 470
ipv6_unicast_enable
 vtss_l3_rleg_conf_t, 160
ipv6uc_received_frames
 vtss_l3_counters_t, 157
ipv6uc_received_octets
 vtss_l3_counters_t, 157
ipv6uc_transmitted_frames
 vtss_l3_counters_t, 158
ipv6uc_transmitted_octets
 vtss_l3_counters_t, 158
irq_trigger
 vtss_acl_action_t, 59
is_high_amp
 vtss_phy_10g_clk_src_t, 238
is_host
 vtss_phy_10g_ib_conf_t, 246
 vtss_phy_10g_ob_status_t, 276
is_host_side
 vtss_phy_10g_auto_failover_conf_t, 219
is_host_wan
 vtss_phy_10g_mode_t, 271
is_init
 vtss_phy_10g_mode_t, 273
isdx
 vtss_packet_rx_info_t, 192
 vtss_packet_tx_info_t, 213
 vtss_vstax_rx_header_t, 533
isdx_dont_use
 vtss_packet_tx_info_t, 214
ivid
 vtss_evc_pb_conf_t, 95
keep_ttl
 vtss_vstax_tx_header_t, 535
key
 vtss_ece_t, 83
 vtss_qce_t, 453
 vtss_vce_t, 515
l
 vtss_phy_10g_ib_conf_t, 244
l2_types.h
 vtss_sfflow_type_t, 537
l3_other_queue
 vtss_packet_rx_queue_map_t, 201
l3_uc_queue
 vtss_packet_rx_queue_map_t, 201
l_apc_conf
 vtss_phy_10g_mode_t, 273
l_clk_src
 vtss_phy_10g_mode_t, 271
l_h
 vtss_phy_10g_clause_37_control_t, 235
l_ib_conf
 vtss_phy_10g_mode_t, 272
l_media
 vtss_phy_10g_mode_t, 272
l_offset_guard
 vtss_phy_10g_mode_t, 269
l_pcsc
 vtss_phy_10g_serdes_status_t, 300
l_pll5g_fsm_lock
 vtss_phy_10g_serdes_status_t, 296
l_pll5g_fsm_stat
 vtss_phy_10g_serdes_status_t, 296
l_pll5g_gain
 vtss_phy_10g_serdes_status_t, 296
l_pll5g_lock_status
 vtss_phy_10g_serdes_status_t, 295
l_pma
 vtss_phy_10g_serdes_status_t, 300
l_rx_rcpll_fsm_status

vtss_phy_10g_serd़es_status_t, 298
 l_rx_rcpll_lock_status
 vtss_phy_10g_serd़es_status_t, 297
 l_rx_rcpll_range
 vtss_phy_10g_serd़es_status_t, 297
 l_rx_rcpll_vco_load
 vtss_phy_10g_serd़es_status_t, 297
 l_tx_rcpll_fsm_status
 vtss_phy_10g_serd़es_status_t, 299
 l_tx_rcpll_lock_status
 vtss_phy_10g_serd़es_status_t, 299
 l_tx_rcpll_range
 vtss_phy_10g_serd़es_status_t, 299
 l_tx_rcpll_vco_load
 vtss_phy_10g_serd़es_status_t, 299
 latch_timestamp
 vtss_packet_tx_info_t, 212
 layer
 vtss_debug_info_t, 68
 lb_type
 vtss_phy_10g_loopback_t, 261
 vtss_phy_api.h, 915
 ld
 vtss_phy_10g_ib_conf_t, 245
 ld_abil
 vtss_phy_10g_base_kr_train_aneg_t, 227
 ld_pre_init
 vtss_phy_10g_base_kr_training_t, 229
 leaf
 vtss_evc_pb_conf_t, 96
 leaf_ivid
 vtss_evc_pb_conf_t, 95
 leaf_vid
 vtss_evc_pb_conf_t, 95
 learn
 vtss_acl_action_t, 58
 vtss_packet_rx_header_t, 182
 learn_queue
 vtss_packet_rx_queue_map_t, 200
 learned
 vtss_mac_table_status_t, 169
 learning
 vtss_evc_conf_t, 91
 vtss_policer_ext_t, 404
 vtss_vlan_vid_conf_t, 525
 length
 vtss_ace_frame_arp_t, 38
 vtss_packet_rx_header_t, 182
 vtss_packet_rx_info_t, 185
 vtss_packet_rx_meta_t, 196
 vtss_packet_tx_ifh_t, 205
 vtss_phy_veriphy_result_t, 399
 level
 vtss_phy_power_status_t, 335
 vtss_policer_t, 406
 vtss_shaper_t, 478
 vtss_trace_conf_t, 486
 levn
 vtss_phy_10g_jitter_conf_t, 252
 vtss_phy_10g_ob_status_t, 275
 lfault
 vtss_phy_10g_pkt_mon_conf_t, 283
 limit_cpu_traffic
 vtss_policer_ext_t, 405
 limit_noncpu_traffic
 vtss_policer_ext_t, 405
 line
 vtss_api_lock_t, 64
 vtss_phy_10g_clause_37_cmn_status_t, 234
 vtss_phy_10g_clause_37_control_t, 235
 vtss_phy_10g_prbs_gen_conf_t, 286
 vtss_phy_10g_prbs_mon_conf_t, 287
 vtss_phy_10g_vsphere_conf_t, 306
 vtss_phy_10g_vsphere_scan_conf_t, 307
 line_kr
 vtss_phy_10g_base_kr_train_aneg_t, 227
 line_rate
 vtss_dlb_policer_conf_t, 72
 line_rx
 vtss_phy_10g_polarity_inv_t, 284
 line_rx_force
 vtss_ewis_force_mode_s, 112
 line_tx
 vtss_phy_10g_polarity_inv_t, 284
 line_tx_force
 vtss_ewis_force_mode_s, 112
 link
 vtss_phy_10g_clause_37_status_t, 236
 vtss_phy_veriphy_result_t, 399
 vtss_port_sgmii_aneg_t, 437
 vtss_port_status_t, 439
 link_6g_distance
 vtss_phy_10g_mode_t, 271
 link_down
 vtss_port_status_t, 439
 vtss_sublayer_status_t, 479
 link_stat
 vtss_ewis_status_s, 126
 llabs
 vtss_os_ecos.h, 761
 llc
 vtss_ace_frame_llc_t, 50
 vtss_ace_t, 54
 vtss_qce_key_t, 450
 vtss_vce_key_t, 512
 lock_status
 vtss_lcpll_status_t, 163
 locked
 vtss_mac_table_entry_t, 167
 loop
 vtss_port_conf_t, 414
 loopback
 vtss_ewis_test_conf_s, 127
 lopc_stat
 vtss_phy_10g_status_t, 304
 low

vtss_phy_timestamp_t, 342
vtss_vcap_udp_tcp_t, 500
vtss_vcap_vr_t, 503
low_duty_cycle
 vtss_phy_ltc_freq_synth_s, 326
lower
 vtss_phy_ts_eth_conf_t, 355
 vtss_phy_ts_mpls_lvr_rng_t, 379
 vtss_phy_ts_ptp_conf_t, 387
 vtss_phy_ts_y1731_oam_conf_t, 396
lp_advertisement
 vtss_eee_port_conf_t, 87
lp_aneg_able
 vtss_phy_10g_kr_status_aneg_t, 255
lpcs_1g
 vtss_phy_10g_status_t, 303
lref_for_host
 vtss_phy_10g_mode_t, 271
lrn_all_queue
 vtss_packet_rx_queue_map_t, 201
ls_inv
 vtss_phy_ts_sertod_conf_t, 392
ls_lpbk
 vtss_phy_ts_alt_clock_mode_s, 346
ls_pps_lpbk
 vtss_phy_ts_alt_clock_mode_s, 346

MAC_ADDR_BROADCAST
 types.h, 589

MAX_CFG_BUF_SIZE
 vtss_phy_api.h, 896

MAX_STAT_BUF_SIZE
 vtss_phy_api.h, 896

MAX_WOL_MAC_ADDR_SIZE
 vtss_phy_api.h, 907

MAX_WOL_PASSWD_SIZE
 vtss_phy_api.h, 907

MIN_WOL_PASSWD_SIZE
 vtss_phy_api.h, 907

mac
 vtss_ece_key_t, 79
 vtss_qce_key_t, 449
 vtss_vce_key_t, 511
 vtss_vid_mac_t, 518

mac_addr
 vtss_phy_ts_eth_conf_t, 354

mac_if
 vtss_phy_reset_conf_t, 336

mac_if_pcs
 vtss_phy_conf_t, 317

mac_serdes_equipment_enable
 vtss_phy_loopback_t, 325

mac_serdes_facility_enable
 vtss_phy_loopback_t, 324

mac_serdes_input_enable
 vtss_phy_loopback_t, 324

mac_vid_queue
 vtss_packet_rx_queue_map_t, 200

macsec_ena

 vtss_phy_ts_init_conf_t, 370
magic_pkt_cnt
 vtss_phy_wol_conf_t, 401

main_status
 vtss_phy_10g_prbs_mon_conf_t, 289

map
 vtss_packet_rx_conf_t, 181

mask
 vtss_phy_ts_ach_conf_t, 344
 vtss_phy_ts_eth_conf_t, 356, 357
 vtss_phy_ts_gen_conf_t, 363
 vtss_phy_ts_generic_action_t, 365
 vtss_phy_ts_ip_conf_t, 374
 vtss_phy_ts_ptp_conf_t, 386
 vtss_phy_ts_y1731_oam_conf_t, 396
 vtss_vcap_ip_t, 492
 vtss_vcap_u128_t, 493
 vtss_vcap_u16_t, 494
 vtss_vcap_u24_t, 495
 vtss_vcap_u32_t, 496
 vtss_vcap_u40_t, 497
 vtss_vcap_u48_t, 498
 vtss_vcap_u8_t, 499
 vtss_vcap_vid_t, 502
 vtss_vcap_vr_t, 503

masquerade_port
 vtss_packet_tx_info_t, 215

master
 vtss_phy_10g_mode_t, 270
 vtss_phy_conf_1g_t, 315
 vtss_phy_status_1g_t, 341

master_cfg_fault
 vtss_phy_status_1g_t, 341

match_mode
 vtss_phy_ts_ip_conf_t, 374

max
 ib_par_cfg, 32

max_bist_frames
 vtss_phy_10g_prbs_mon_conf_t, 287

max_frame_length
 vtss_port_conf_t, 412

max_length
 port_custom_conf_t, 34

max_prob_1
 vtss_red_t, 468

max_prob_2
 vtss_red_t, 468

max_prob_3
 vtss_red_t, 468

max_tags
 port_custom_conf_t, 34
 vtss_port_conf_t, 413

max_th
 vtss_red_t, 467

mc_no_flood
 vtss_policer_ext_t, 404

mdi
 vtss_phy_conf_t, 316

mdi_cross
 vtss_port_status_t, 441

media_if
 vtss_phy_reset_conf_t, 336

media_if_pcs
 vtss_phy_conf_t, 317

media_mac_serdes_crc
 vtss_phy_statistic_t, 340

media_mac_serdes_good
 vtss_phy_statistic_t, 340

media_serdes_equipment_enable
 vtss_phy_loopback_t, 325

media_serdes_facility_enable
 vtss_phy_loopback_t, 325

media_serdes_input_enable
 vtss_phy_loopback_t, 325

meg_level
 vtss_phy_ts_y1731_oam_conf_t, 396

miim_addr
 vtss_port_map_t, 427

miim_chip_no
 vtss_port_map_t, 427

miim_controller
 vtss_port_map_t, 427

miim_read
 vtss_init_conf_t, 142

miim_write
 vtss_init_conf_t, 142

min
 ib_par_cfg, 31

min_th
 vtss_red_t, 467

mirror
 vtss_vlan_vid_conf_t, 525
 vtss_vstax_port_conf_t, 529

mld_cpu_only
 vtss_packet_rx_reg_t, 204

mmd_read
 vtss_init_conf_t, 142

mmd_read_inc
 vtss_init_conf_t, 143

mmd_write
 vtss_init_conf_t, 143

mode
 vtss_ewis_tti_s, 129
 vtss_gpio_10g_gpio_mode_t, 139
 vtss_phy_10g_ckout_conf_t, 230
 vtss_phy_10g_host_clk_conf_t, 242
 vtss_phy_10g_line_clk_conf_t, 260
 vtss_phy_10g_rxckout_conf_t, 291
 vtss_phy_10g_sckout_conf_t, 292
 vtss_phy_10g_txckout_conf_t, 305
 vtss_phy_conf_t, 316
 vtss_phy_led_mode_select_t, 323
 vtss_phy_power_conf_t, 334
 vtss_qs_conf_t, 464
 vtss_sgpio_port_conf_t, 476
 vtss_ts_operation_mode_t, 489

moved
 vtss_mac_table_status_t, 170

mpls_opt
 vtss_phy_ts_oam_engine_flow_conf_t, 384
 vtss_phy_ts_ptp_engine_flow_conf_t, 391

msg_type
 vtss_phy_ts_fifo_sig_t, 360

multicast
 vtss_policer_ext_t, 403

mux_mode
 vtss_init_conf_t, 144

n_ebc_thr_l
 vtss_ewis_counter_threshold_s, 104

n_ebc_thr_p
 vtss_ewis_counter_threshold_s, 105

n_ebc_thr_s
 vtss_ewis_counter_threshold_s, 104

nanoseconds
 vtss_phy_timestamp_t, 343
 vtss_timestamp_t, 485

near_end_enable
 vtss_phy_loopback_t, 324

network
 vtss_evc_conf_t, 92
 vtss_ipv4_uc_t, 149
 vtss_ipv6_uc_t, 152

next_page
 vtss_phy_10g_clause_37_adv_t, 233
 vtss_port_clause_37_adv_t, 409

next_prot_offset
 vtss_phy_ts_gen_conf_t, 362

nni
 vtss_evc_pb_conf_t, 95

no_of_errors
 vtss_phy_10g_prbs_mon_conf_t, 288

no_sync
 vtss_phy_10g_prbs_mon_conf_t, 290

no_wait
 vtss_packet_rx_meta_t, 194

now
 vtss_mtimer_t, 173

npi
 vtss_packet_rx_queue_conf_t, 199

num_tag
 vtss_phy_ts_eth_conf_t, 354

number
 vtss_phy_led_mode_select_t, 323

oam
 vtss_phy_ts_engine_flow_conf_t, 351

oam_conf
 vtss_phy_ts_engine_action_t, 349
 vtss_phy_ts_oam_engine_action_t, 383

oam_info
 vtss_packet_rx_info_t, 191

oam_info_decoded
 vtss_packet_rx_info_t, 192

oam_type

vtss_packet_tx_info_t, 213
ob_conf
 vtss_phy_10g_mode_t, 266
ob_post0
 serdes_fields_t, 36
ob_sr
 serdes_fields_t, 36
obey
 vtss_port_flow_control_conf_t, 420
obey_pause
 vtss_aneg_t, 63
offs
 vtss_phy_10g_ib_conf_t, 243
one_pps_mode
 vtss_ts_ext_clock_mode_t, 486
one_step_txfifo
 vtss_phy_ts_init_conf_t, 371
op_enable
 vtss_phy_ts_sertod_conf_t, 392
op_mode
 vtss_phy_10g_apc_conf_t, 216
op_mode_flag
 vtss_phy_10g_apc_conf_t, 217
oper_mode
 vtss_phy_10g_mode_t, 263
oper_mode_t
 vtss_phy_10g_api.h, 822
oper_up
 port_custom_conf_t, 35
optimized_for_power
 vtss_eee_port_conf_t, 87
options
 vtss_ace_frame_ipv4_t, 41
 vtss_vce_frame_ipv4_t, 507
options.h
 VIPER_B_FIFO_RESET, 553
 VTSS_ARCH_JAGUAR_1, 540
 VTSS_ARCH_JAGUAR_1_CE_SWITCH, 540
 VTSS_ARCH_MALIBU_B, 556
 VTSS_ARCH_MALIBU, 555
 VTSS_ARCH_VENICE_C, 556
 VTSS_CHIP_10G_PHY, 542
 VTSS_CHIP CU PHY, 554
 VTSS_FEATURE_10GBASE_KR, 552
 VTSS_FEATURE_10G, 543
 VTSS_FEATURE_ACL_V1, 551
 VTSS_FEATURE_ACL, 551
 VTSS_FEATURE_AGGR_GLAG, 541
 VTSS_FEATURE_CLAUSE_37, 543
 VTSS_FEATURE_E_TREE, 540
 VTSS_FEATURE_EDC_FW_LOAD, 555
 VTSS_FEATURE_EEE, 552
 VTSS_FEATURE_EVC, 540
 VTSS_FEATURE_EXC_COL_CONT, 544
 VTSS_FEATURE_FAN, 541, 551
 VTSS_FEATURE_IRQ_CONTROL, 552
 VTSS_FEATURE_LAYER2, 549
 VTSS_FEATURE_LAYER3, 550
 VTSS_FEATURE_LED_POW_REDUC, 552
 VTSS_FEATURE_MAC_AGE_AUTO, 550
 VTSS_FEATURE_MAC_CPU_QUEUE, 550
 VTSS_FEATURE_MISC, 542
 VTSS_FEATURE_NPI, 543
 VTSS_FEATURE_PACKET_GROUPING, 548
 VTSS_FEATURE_PACKET_PORT_REG, 549
 VTSS_FEATURE_PACKET_RX, 548
 VTSS_FEATURE_PACKET_TX, 548
 VTSS_FEATURE_PACKET, 548
 VTSS_FEATURE_PHY_TIMESTAMP, 540
 VTSS_FEATURE_PORT_CNT_BRIDGE, 544
 VTSS_FEATURE_PORT_CNT_ETHER_LIKE, 544
 VTSS_FEATURE_PORT_CONTROL, 543
 VTSS_FEATURE_PORT_IFH, 543
 VTSS_FEATURE_PORT_MUX, 550
 VTSS_FEATURE_PVLAN, 542
 VTSS_FEATURE_QCL_V2, 545
 VTSS_FEATURE_QCL, 544
 VTSS_FEATURE_QOS_CLASSIFICATION_V2, 547
 VTSS_FEATURE_QOS_DSCP_REMARK_V2, 547
 VTSS_FEATURE_QOS_DSCP_REMARK, 547
 VTSS_FEATURE_QOS_EGRESS_QUEUE_SH APERS_EB, 547
 VTSS_FEATURE_QOS_EGRESS_QUEUE_SH_APERS, 547
 VTSS_FEATURE_QOS_POLICER_DLDB, 553
 VTSS_FEATURE_QOS_PORT_POLICER_EXT DPBL, 545
 VTSS_FEATURE_QOS_PORT_POLICER_EXT FPS, 545
 VTSS_FEATURE_QOS_PORT_POLICER_EXT FC, 545
 VTSS_FEATURE_QOS_PORT_POLICER_EXT TTM, 546
 VTSS_FEATURE_QOS_PORT_POLICER_EXT, 545
 VTSS_FEATURE_QOS_QUEUE_POLICER, 546
 VTSS_FEATURE_QOS_QUEUE_TX, 546
 VTSS_FEATURE_QOS_SCHEDULER_V2, 546
 VTSS_FEATURE_QOS_TAG_REMARK_V2, 546
 VTSS_FEATURE_QOS_WRED, 548
 VTSS_FEATURE_QOS, 544
 VTSS_FEATURE_SERDES_MACRO_SETTINGS, 552
 VTSS_FEATURE_SERIAL_GPIO, 542
 VTSS_FEATURE_SFLOW, 553
 VTSS_FEATURE_SYNC_10G, 555
 VTSS_FEATURE_SYNC, 551
 VTSS_FEATURE_TIMESTAMP, 540
 VTSS_FEATURE_VCAP, 550, 556
 VTSS_FEATURE_VCL, 551
 VTSS_FEATURE_VLAN_PORT_V2, 549
 VTSS_FEATURE_VLAN_SVL, 549
 VTSS_FEATURE_VLAN_TRANSLATION, 553

VTSS FEATURE VLAN TX TAG, [549](#)
 VTSS FEATURE VSTAX V2, [541](#)
 VTSS FEATURE VSTAX, [541](#)
 VTSS FEATURE WARM START, [554](#), [555](#)
 VTSS FEATURE WIS, [555](#)
 VTSS_OPT_TRACE, [554](#)
 VTSS_OPT_TS_SPI_FPGA, [542](#)
 VTSS_OPT_VAUI_EQ_CTRL, [554](#)
 VTSS_PHY_10G_FIFO_SYNC, [553](#)
 VTSS_PHY_OPT_VERIPHYS, [554](#)
 VTSS_PHY_TS_SPI_CLK_THRU_PPS0, [541](#)
 out_of_range
 vtss_rcpll_status_t, [466](#)
 outer_tag
 vtss_ece_action_t, [74](#)
 vtss_phy_ts_eth_conf_t, [356](#)
 outer_tag_type
 vtss_phy_ts_eth_conf_t, [355](#)
 oversubscription
 vtss_qs_conf_t, [464](#)
 p_gpio
 vtss_gpio_10g_gpio_mode_t, [140](#)
 p_gpio_intrpt
 vtss_gpio_10g_gpio_mode_t, [141](#)
 PHASE_POINTS
 vtss_phy_10g_api.h, [808](#)
 PORT_CAP_100M_FDX
 port.h, [561](#)
 PORT_CAP_100M_HDX
 port.h, [561](#)
 PORT_CAP_10G_FDX
 port.h, [562](#)
 PORT_CAP_10M_FDX
 port.h, [561](#)
 PORT_CAP_10M_HDX
 port.h, [561](#)
 PORT_CAP_1G_FDX
 port.h, [562](#)
 PORT_CAP_1G_PHY
 port.h, [566](#)
 PORT_CAP_2_5G_FDX
 port.h, [562](#)
 PORT_CAP_2_5G_TRI_SPEED_COPPER
 port.h, [570](#)
 PORT_CAP_2_5G_TRI_SPEED_FDX
 port.h, [569](#)
 PORT_CAP_2_5G_TRI_SPEED
 port.h, [569](#)
 PORT_CAP_5G_FDX
 port.h, [562](#)
 PORT_CAP_ANY_FIBER
 port.h, [567](#)
 PORT_CAP_AUTONEG
 port.h, [561](#)
 PORT_CAP_COPPER
 port.h, [563](#)
 PORT_CAP_DUAL_COPPER_100FX
 port.h, [565](#)
 PORT_CAP_DUAL_COPPER
 port.h, [563](#)
 PORT_CAP_DUAL_COPPER_1000X
 port.h, [568](#)
 PORT_CAP_DUAL_FIBER_100FX
 port.h, [564](#)
 PORT_CAP_DUAL_FIBER
 port.h, [563](#)
 PORT_CAP_DUAL_SFP_DETECT
 port.h, [565](#)
 PORT_CAP_FIBER
 port.h, [563](#)
 PORT_CAP_FLOW_CTRL
 port.h, [562](#)
 PORT_CAP_HDX
 port.h, [566](#)
 PORT_CAP_NONE
 port.h, [561](#)
 PORT_CAP_SD_ENABLE
 port.h, [563](#)
 PORT_CAP_SD_HIGH
 port.h, [564](#)
 PORT_CAP_SD_INTERNAL
 port.h, [564](#)
 PORT_CAP_SFP_1G
 port.h, [569](#)
 PORT_CAP_SFP_2_5G
 port.h, [569](#)
 PORT_CAP_SFP_DETECT
 port.h, [565](#)
 PORT_CAP_SFP_ONLY
 port.h, [565](#)
 PORT_CAP_SFP_SD_HIGH
 port.h, [569](#)
 PORT_CAP_SPEED_DUAL_ANY_FIBER_FIXED_SPEED
 port.h, [568](#)
 PORT_CAP_SPEED_DUAL_ANY_FIBER
 port.h, [568](#)
 PORT_CAP_STACKING
 port.h, [565](#)
 PORT_CAP_TRI_SPEED_COPPER
 port.h, [566](#)
 PORT_CAP_TRI_SPEED_DUAL_ANY_FIBER_FIXESPEED
 port.h, [568](#)
 PORT_CAP_TRI_SPEED_DUAL_ANY_FIBER
 port.h, [568](#)
 PORT_CAP_TRI_SPEED_DUAL_COPPER
 port.h, [567](#)
 PORT_CAP_TRI_SPEED_DUAL_FIBER_100FX
 port.h, [567](#)
 PORT_CAP_TRI_SPEED_DUAL_FIBER
 port.h, [567](#)
 PORT_CAP_TRI_SPEED_FDX
 port.h, [566](#)
 PORT_CAP_TRI_SPEED_FIBER
 port.h, [567](#)

PORT_CAP_TRI_SPEED
 port.h, 566

PORT_CAP_VTSS_10G_PHY
 port.h, 564

PORT_CAP_XAUI_LANE_FLIP
 port.h, 564

PRBS_status
 vtss_phy_10g_prbs_mon_conf_t, 289

PRli64
 types.h, 579

PRlu64
 types.h, 578

PRlx64
 types.h, 579

part_number
 vtss_chip_id_t, 66
 vtss_phy_10g_id_t, 249
 vtss_phy_type_t, 397

partner_advertisement
 vtss_phy_10g_clause_37_status_t, 237

passwd
 vtss_secure_on_passwd_t, 472

path_force
 vtss_ewis_force_mode_s, 113

path_txti
 vtss_ewis_conf_s, 100

pb
 vtss_evc_conf_t, 92

pbb_en
 vtss_phy_ts_eth_conf_t, 353

pcp
 vtss_ece_inner_tag_t, 78
 vtss_ece_outer_tag_t, 82
 vtss_ece_tag_t, 85
 vtss_mirror_conf_t, 171
 vtss_qce_tag_t, 454
 vtss_vce_tag_t, 516
 vtss_vlan_tag_t, 522

pcp_dei_preserve
 vtss_ece_inner_tag_t, 77
 vtss_ece_outer_tag_t, 82

pcs
 vtss_phy_10g_cnt_t, 238
 vtss_phy_10g_status_t, 302

pd_enable
 vtss_phy_mac_serdes_pcs_ctrl_t, 328

pdu_offset
 vtss_packet_tx_info_t, 215

perf_mode
 vtss_ewis_conf_s, 101

pf_ebc_l
 vtss_ewis_counter_s, 103
 vtss_ewis_perf_s, 118

pf_ebc_mode_l
 vtss_ewis_perf_mode_s, 117

pf_ebc_mode_p
 vtss_ewis_perf_mode_s, 117

pf_ebc_p
 vtss_ewis_counter_s, 103
 vtss_ewis_perf_s, 119

pfc
 port_custom_conf_t, 33
 vtss_port_flow_control_conf_t, 420

phy.h
 VTSS_PHY_POWER_ACTIPHYS_BIT, 557
 VTSS_PHY_POWER_DYNAMIC_BIT, 557
 vtss_phy_power_mode_t, 557
 vtss_phy_veriphy_status_t, 558

phy_api_base_no
 vtss_phy_10g_id_t, 250
 vtss_phy_type_t, 398

pi
 vtss_init_conf_t, 144

pkt_len
 vtss_phy_10g_pkt_gen_conf_t, 280

pkt_mode
 vtss_phy_reset_conf_t, 337

pma
 vtss_phy_10g_status_t, 302

pma_txratecontrol
 vtss_phy_10g_mode_t, 269

pn_ebc_l
 vtss_ewis_counter_s, 103
 vtss_ewis_perf_s, 118

pn_ebc_mode_l
 vtss_ewis_perf_mode_s, 117

pn_ebc_mode_p
 vtss_ewis_perf_mode_s, 117

pn_ebc_mode_s
 vtss_ewis_perf_mode_s, 116

pn_ebc_p
 vtss_ewis_counter_s, 103
 vtss_ewis_perf_s, 119

pn_ebc_s
 vtss_ewis_counter_s, 103
 vtss_ewis_perf_s, 118

polarity
 vtss_phy_10g_mode_t, 270

police
 vtss_acl_action_t, 57

policer_ext_port
 vtss_qos_port_conf_t, 459

policer_id
 vtss_ece_action_t, 74
 vtss_evc_conf_t, 91

policer_no
 vtss_acl_action_t, 58

policer_port
 vtss_qos_port_conf_t, 458

policer_queue
 vtss_qos_port_conf_t, 459

policy
 vtss_ace_t, 52

policy_no
 vtss_acl_port_conf_t, 60
 vtss_ece_action_t, 75

vtss_vce_action_t, 505
pop_tag
 vtss_ece_action_t, 74
port
 vtss_gpio_10g_gpio_mode_t, 139
 vtss_qs_conf_t, 465
port.h
 PORT_CAP_100M_FDX, 561
 PORT_CAP_100M_HDX, 561
 PORT_CAP_10G_FDX, 562
 PORT_CAP_10M_FDX, 561
 PORT_CAP_10M_HDX, 561
 PORT_CAP_1G_FDX, 562
 PORT_CAP_1G_PHY, 566
 PORT_CAP_2_5G_FDX, 562
 PORT_CAP_2_5G_TRI_SPEED_COPPER, 570
 PORT_CAP_2_5G_TRI_SPEED_FDX, 569
 PORT_CAP_2_5G_TRI_SPEED, 569
 PORT_CAP_5G_FDX, 562
 PORT_CAP_ANY_FIBER, 567
 PORT_CAP_AUTONEG, 561
 PORT_CAP_COPPER, 563
 PORT_CAP_DUAL_COPPER_100FX, 565
 PORT_CAP_DUAL_COPPER, 563
 PORT_CAP_DUAL_FIBER_1000X, 568
 PORT_CAP_DUAL_FIBER_100FX, 564
 PORT_CAP_DUAL_FIBER, 563
 PORT_CAP_DUAL_SFP_DETECT, 565
 PORT_CAP_FIBER, 563
 PORT_CAP_FLOW_CTRL, 562
 PORT_CAP_HDX, 566
 PORT_CAP_NONE, 561
 PORT_CAP_SD_ENABLE, 563
 PORT_CAP_SD_HIGH, 564
 PORT_CAP_SD_INTERNAL, 564
 PORT_CAP_SFP_1G, 569
 PORT_CAP_SFP_2_5G, 569
 PORT_CAP_SFP_DETECT, 565
 PORT_CAP_SFP_ONLY, 565
 PORT_CAP_SFP_SD_HIGH, 569
 PORT_CAP_SPEED_DUAL_ANY_FIBER_FIXE←
 D_SPEED, 568
 PORT_CAP_SPEED_DUAL_ANY_FIBER, 568
 PORT_CAP_STACKING, 565
 PORT_CAP_TRI_SPEED_COPPER, 566
 PORT_CAP_TRI_SPEED_DUAL_ANY_FIBER←
 FIXED_SFP_SPEED, 568
 PORT_CAP_TRI_SPEED_DUAL_ANY_FIBER,
 568
 PORT_CAP_TRI_SPEED_DUAL_COPPER, 567
 PORT_CAP_TRI_SPEED_DUAL_FIBER_100FX,
 567
 PORT_CAP_TRI_SPEED_DUAL_FIBER, 567
 PORT_CAP_TRI_SPEED_FDX, 566
 PORT_CAP_TRI_SPEED_FIBER, 567
 PORT_CAP_TRI_SPEED, 566
 PORT_CAP_VTSS_10G_PHY, 564
 PORT_CAP_XAUI_LANE_FLIP, 564
 port_cap_t, 570
 vtss_fiber_port_speed_t, 571
 vtss_port_speed_t, 570
port_0
 vtss_vstax_conf_t, 527
port_1
 vtss_vstax_conf_t, 527
port_cap_t
 port.h, 570
port_cnt
 vtss_phy_type_t, 398
port_conf
 vtss_sgpi_conf_t, 475
port_custom_conf_t, 32
 adv_dis, 34
 autoneg, 33
 dual_media_fiber_speed, 34
 enable, 33
 exc_col_cont, 34
 fdx, 33
 flow_control, 33
 frame_length_chk, 35
 max_length, 34
 max_tags, 34
 oper_up, 35
 pfc, 33
 speed, 33
port_forward
 vtss_acl_action_t, 58
port_list
 vtss_debug_info_t, 68
 vtss_ece_key_t, 79
 vtss_qce_key_t, 449
 vtss_vce_key_t, 511
port_mask
 vtss_ts_timestamp_alloc_t, 490
port_max
 vtss_qs_conf_t, 464
port_min
 vtss_qs_conf_t, 464
port_no
 vtss_ace_t, 52
 vtss_acl_action_t, 58
 vtss_eps_port_conf_t, 90
 vtss_mirror_conf_t, 171
 vtss_npi_conf_t, 174
 vtss_packet_frame_info_t, 176
 vtss_packet_port_info_t, 179
 vtss_packet_rx_header_t, 182
 vtss_packet_rx_info_t, 185
 vtss_phy_10g_auto_failover_conf_t, 219
 vtss_sync_clock_in_t, 480
 vtss_vstax_rx_header_t, 532
 vtss_vstax_tx_header_t, 534
port_tx
 vtss_packet_frame_info_t, 177
port_type
 vtss_vlan_port_conf_t, 520

ports
 vtss_vlan_trans_port2grp_conf_t, 524

power_down
 vtss_port_conf_t, 412

ppr
 vtss_fan_conf_t, 138

pps_ls_lpbk
 vtss_phy_ts_alt_clock_mode_s, 346

pps_offset
 vtss_phy_ts_pps_config_s, 385

pps_output_enable
 vtss_phy_ts_pps_config_s, 385

pps_width_adj
 vtss_phy_ts_pps_config_s, 385

pr
 vtss_phy_10g_fifo_sync_t, 239

prbs
 vtss_phy_10g_ib_conf_t, 245

prbs_check_input_invert
 vtss_phy_10g_prbs_mon_conf_t, 288

prbs_gen
 vtss_phy_10g_pcs_prbs_gen_conf_t, 277

prbs_inv
 vtss_phy_10g_ib_conf_t, 245

prbs_mon
 vtss_phy_10g_pcs_prbs_mon_conf_t, 277

prbsn_sel
 vtss_phy_10g_prbs_mon_conf_t, 288

prbsn_tx_io
 vtss_phy_10g_prbs_gen_conf_t, 286

prbsn_tx_iw
 vtss_phy_10g_prbs_gen_conf_t, 286

prbsn_tx_sel
 vtss_phy_10g_prbs_gen_conf_t, 285

prefix_size
 vtss_ip_network_t, 148
 vtss_ipv4_network_t, 149
 vtss_ipv6_network_t, 151

prev_version
 vtss_restart_status_t, 469

prio
 vtss_qce_action_t, 442
 vtss_vstax_tx_header_t, 534

prio_enable
 vtss_qce_action_t, 442

prios
 vtss_qos_conf_t, 456

prop
 vtss_port_counters_t, 415

proto
 vtss_ace_frame_ipv4_t, 42
 vtss_ace_frame_ipv6_t, 46
 vtss_qce_frame_ipv4_t, 445
 vtss_qce_frame_ipv6_t, 446
 vtss_vce_frame_ipv4_t, 507
 vtss_vce_frame_ipv6_t, 508

proto_id
 vtss_phy_ts_ach_conf_t, 345

ptp
 vtss_phy_10g_pkt_gen_conf_t, 279
 vtss_phy_ts_engine_flow_conf_t, 351

ptp_action
 vtss_packet_tx_info_t, 211

ptp_conf
 vtss_phy_ts_engine_action_t, 349
 vtss_phy_ts_ptp_engine_action_t, 388

ptp_id
 vtss_packet_tx_info_t, 211

ptp_timestamp
 vtss_packet_tx_info_t, 212

ptp_ts_ns
 vtss_phy_10g_pkt_gen_conf_t, 281

ptp_ts_sec
 vtss_phy_10g_pkt_gen_conf_t, 280

pvid
 vtss_vlan_port_conf_t, 520

qos_class_map
 vtss_qos_port_conf_t, 461

qs_conf
 vtss_init_conf_t, 145

qsgmii
 vtss_serdes_macro_conf_t, 473

queue
 vtss_packet_rx_conf_t, 180

queue_mask
 vtss_packet_rx_header_t, 182

queue_max
 vtss_qs_conf_t, 465

queue_min
 vtss_qs_conf_t, 465

queue_no
 vtss_vstax_tx_header_t, 535

queue_pct
 vtss_qos_port_conf_t, 463

r
 vtss_vcap_vr_t, 504

r_ctrl
 vtss_phy_10g_ob_status_t, 274

range
 vtss_phy_ts_eth_conf_t, 356
 vtss_phy_ts_ptp_conf_t, 387
 vtss_phy_ts_y1731_oam_conf_t, 396

range_en
 vtss_phy_ts_ptp_conf_t, 386
 vtss_phy_ts_y1731_oam_conf_t, 395

rate
 vtss_acl_policer_conf_t, 59
 vtss_phy_10g_mode_t, 270
 vtss_policer_t, 406
 vtss_shaper_t, 478

raw_ident
 vtss_irq_status_t, 154

raw_mask
 vtss_irq_status_t, 155

raw_status

vtss_irq_status_t, 155
 rcomp
 vtss_phy_10g_serdes_status_t, 294
 rcvrd_clk
 vtss_phy_10g_mode_t, 265
 rcvrd_clk_div
 vtss_phy_10g_mode_t, 265
 rdil
 vtss_ewis_cons_act_s, 102
 rdil_on_ais_l
 vtss_ewis_rdil_cons_act_s, 120
 rdil_on_lof
 vtss_ewis_rdil_cons_act_s, 120
 rdil_on_lopc
 vtss_ewis_rdil_cons_act_s, 120
 rdil_on_los
 vtss_ewis_rdil_cons_act_s, 120
 recvrd_clk_sel
 vtss_phy_10g_host_clk_conf_t, 242
 vtss_phy_10g_line_clk_conf_t, 260
 red
 vtss_qos_port_conf_t, 458
 reg
 vtss_packet_rx_conf_t, 181
 reg_read
 vtss_init_conf_t, 142
 reg_write
 vtss_init_conf_t, 142
 remote_entry
 vtss_mac_table_entry_t, 168
 remote_fault
 vtss_phy_10g_clause_37_adv_t, 232
 vtss_phy_media_serdes_pcs_ctrl_t, 331
 vtss_port_clause_37_adv_t, 408
 vtss_port_status_t, 440
 replaced
 vtss_mac_table_status_t, 169
 req
 vtss_ace_frame_arp_t, 37
 request_10g
 vtss_phy_10g_kr_status_aneg_t, 254
 request_1g
 vtss_phy_10g_kr_status_aneg_t, 254
 request_fec_change
 vtss_phy_10g_kr_status_aneg_t, 254
 reset
 vtss_phy_10g_apc_status_t, 217
 vtss_phy_10g_pkt_mon_conf_t, 282
 restart
 vtss_phy_mac_serdes_pcs_ctrl_t, 328
 vtss_restart_status_t, 469
 restart_info_port
 vtss_init_conf_t, 144
 restart_info_src
 vtss_init_conf_t, 144
 revision
 vtss_chip_id_t, 66
 vtss_phy_10g_id_t, 249
 vtss_phy_type_t, 397
 rgmii
 vtss_phy_reset_conf_t, 336
 rgmii_skew_delay_psec_t
 vtss_phy_api.h, 910
 rleg_mode
 vtss_l3_common_conf_t, 156
 rmon
 vtss_port_counters_t, 415
 route
 vtss_routing_entry_t, 471
 routing_enable
 vtss_l3_common_conf_t, 156
 rx_ch
 vtss_phy_10g_lane_sync_conf_t, 259
 rx_clk_skew_ps
 vtss_phy_rgmii_conf_t, 338
 rx_discard
 vtss_evc_counters_t, 93
 rx_err_cnt_base_tx
 vtss_phy_statistic_t, 339
 rx_etherStatsBroadcastPkts
 vtss_port_rmon_counters_t, 430
 rx_etherStatsCRCAlignErrors
 vtss_port_rmon_counters_t, 431
 rx_etherStatsDropEvents
 vtss_port_rmon_counters_t, 430
 rx_etherStatsFragments
 vtss_port_rmon_counters_t, 431
 rx_etherStatsJabbers
 vtss_port_rmon_counters_t, 431
 rx_etherStatsMulticastPkts
 vtss_port_rmon_counters_t, 430
 rx_etherStatsOctets
 vtss_port_rmon_counters_t, 430
 rx_etherStatsOversizePkts
 vtss_port_rmon_counters_t, 431
 rx_etherStatsPkts
 vtss_port_rmon_counters_t, 430
 rx_etherStatsPkts1024to1518Octets
 vtss_port_rmon_counters_t, 433
 rx_etherStatsPkts128to255Octets
 vtss_port_rmon_counters_t, 432
 rx_etherStatsPkts1519toMaxOctets
 vtss_port_rmon_counters_t, 433
 rx_etherStatsPkts256to511Octets
 vtss_port_rmon_counters_t, 432
 rx_etherStatsPkts512to1023Octets
 vtss_port_rmon_counters_t, 432
 rx_etherStatsPkts64Octets
 vtss_port_rmon_counters_t, 432
 rx_etherStatsPkts65to127Octets
 vtss_port_rmon_counters_t, 432
 rx_etherStatsUndersizePkts
 vtss_port_rmon_counters_t, 431
 rx_fault
 vtss_sublayer_status_t, 479
 rx_frames

vtss_basic_counters_t, 65
rx_green
 vtss_evc_counters_t, 93
rx_link
 vtss_sublayer_status_t, 479
rx_macro
 vtss_phy_10g_lane_sync_conf_t, 258
rx_prio
 vtss_port_proprietary_counters_t, 428
rx_red
 vtss_evc_counters_t, 93
rx_ts_len
 vtss_phy_ts_init_conf_t, 369
rx_ts_pos
 vtss_phy_ts_init_conf_t, 368
rx_yellow
 vtss_evc_counters_t, 93
s_etype
 vtss_vlan_conf_t, 519
s_tag
 vtss_vce_tag_t, 517
s_tagged
 vtss_ece_tag_t, 85
sampling_rate
 vtss_sflow_port_conf_t, 474
scan_conf
 vtss_phy_10g_vsphere_scan_status_t, 309
scan_type
 vtss_phy_10g_vsphere_conf_t, 306
sd6g_calib_done
 vtss_phy_10g_mode_t, 273
sd_active_high
 vtss_port_conf_t, 411
sd_enable
 vtss_port_conf_t, 411
sd_internal
 vtss_port_conf_t, 411
sec
 vtss_timeofday_t, 483, 484
sec_msb
 vtss_timestamp_t, 484
seconds
 vtss_phy_timestamp_t, 343
 vtss_timestamp_t, 485
section_cons_act
 vtss_ewis_conf_s, 99
section_txti
 vtss_ewis_conf_s, 99
secure_on_enable
 vtss_phy_wol_conf_t, 400
select
 vtss_eee_port_state_t, 89
seq_zero
 vtss_ace_frame_ipv4_t, 45
 vtss_ace_frame_ipv6_t, 49
sequence_id
 vtss_phy_ts_fifo_sig_t, 360
ser_inv

vtss_phy_10g_base_kr_conf_t, 223
ser_led_frame_rate
 vtss_phy_enhanced_led_control_t, 321
ser_led_output_1
 vtss_phy_enhanced_led_control_t, 320
ser_led_output_2
 vtss_phy_enhanced_led_control_t, 321
ser_led_select
 vtss_phy_enhanced_led_control_t, 321
serdes
 vtss_init_conf_t, 145
 vtss_port_conf_t, 414
serdes1g_vdd
 vtss_serdes_macro_conf_t, 472
serdes6g_vdd
 vtss_serdes_macro_conf_t, 472
serdes_aneg_ena
 vtss_phy_mac_serd_pcs_cntl_t, 329
serdes_conf
 vtss_phy_10g_mode_t, 269
serdes_fields_t, 35
 ob_post0, 36
 ob_sr, 36
serdes_pol_inv_in
 vtss_phy_mac_serd_pcs_cntl_t, 329
 vtss_phy_media_serd_pcs_cntl_t, 331
serdes_pol_inv_out
 vtss_phy_mac_serd_pcs_cntl_t, 329
 vtss_phy_media_serd_pcs_cntl_t, 331
serdes_tx_bad
 vtss_phy_statistic_t, 339
serdes_tx_good
 vtss_phy_statistic_t, 339
sflow_port_no
 vtss_packet_rx_info_t, 191
sflow_queue
 vtss_packet_rx_queue_map_t, 201
sflow_type
 vtss_packet_rx_info_t, 190
sfp_dac
 vtss_port_serdes_conf_t, 436
sgmii_in_pre
 vtss_phy_mac_serd_pcs_cntl_t, 328
sgmii_out_pre
 vtss_phy_mac_serd_pcs_cntl_t, 329
shaper_port
 vtss_qos_port_conf_t, 459
shaper_queue
 vtss_qos_port_conf_t, 459
sig_det
 vtss_phy_10g_ib_status_t, 247
sig_mask
 vtss_phy_ts_fifo_sig_t, 360
sigdet
 vtss_phy_conf_t, 317
sip
 vtss_ace_frame_arp_t, 38
 vtss_ace_frame_ipv4_t, 42

vtss_ace_frame_ipv6_t, 46
 vtss_qce_frame_ipv4_t, 445
 vtss_qce_frame_ipv6_t, 447
 vtss_vce_frame_ipv4_t, 507
 vtss_vce_frame_ipv6_t, 509
 sip_dip_enable
 vtss_aggr_mode_t, 62
 sip_eq_dip
 vtss_ace_frame_ipv4_t, 44
 vtss_ace_frame_ipv6_t, 48
 size
 vtss_packet_rx_queue_conf_t, 198
 skip_rev_check
 vtss_phy_10g_fifo_sync_t, 239
 vtss_phy_ts_fifo_conf_t, 359
 slew
 vtss_phy_10g_ob_status_t, 274
 slewrate
 vtss_phy_10g_base_kr_conf_t, 223
 sm
 vtss_phy_10g_kr_status_aneg_t, 254
 smac
 vtss_ace_frame_arp_t, 37
 vtss_ace_frame_etype_t, 40
 vtss_ace_frame_llc_t, 50
 vtss_ace_frame_snap_t, 51
 vtss_phy_10g_pkt_gen_conf_t, 280
 vtss_port_flow_control_conf_t, 420
 vtss_qce_mac_t, 452
 vtss_vce_mac_t, 514
 smac_enable
 vtss_aggr_mode_t, 61
 smac_match
 vtss_ace_frame_arp_t, 37
 snap
 vtss_ace_frame_snap_t, 51
 vtss_ace_t, 54
 vtss_qce_key_t, 450
 vtss_vce_key_t, 512
 snd_lvl_after_top
 vtss_phy_ts_mpls_conf_t, 377
 snd_lvl_before_end
 vtss_phy_ts_mpls_conf_t, 378
 source
 vtss_gpio_10g_gpio_mode_t, 140
 sp
 vtss_vstax_rx_header_t, 532
 speed
 port_custom_conf_t, 33
 vtss_phy_forced_t, 322
 vtss_port_conf_t, 412
 vtss_port_status_t, 439
 speed_100M
 vtss_port_sgmii_aneg_t, 438
 speed_100m_fdx
 vtss_phy_aneg_t, 312
 speed_100m_hdx
 vtss_phy_aneg_t, 311
 speed_10M
 vtss_port_sgmii_aneg_t, 438
 speed_10m_fdx
 vtss_phy_aneg_t, 311
 speed_10m_hdx
 vtss_phy_aneg_t, 311
 speed_1G
 vtss_port_sgmii_aneg_t, 438
 speed_1g_fdx
 vtss_phy_aneg_t, 312
 speed_1g_hdx
 vtss_phy_aneg_t, 312
 spi_32bit_read_write
 vtss_init_conf_t, 143
 spi_64bit_read_write
 vtss_init_conf_t, 143
 spi_daisy_input
 vtss_phy_daisy_chain_conf_t, 318
 spi_daisy_output
 vtss_phy_daisy_chain_conf_t, 319
 spi_read_write
 vtss_init_conf_t, 143
 sport
 vtss_ace_frame_ipv4_t, 43
 vtss_ace_frame_ipv6_t, 47
 vtss_qce_frame_ipv4_t, 445
 vtss_qce_frame_ipv6_t, 447
 sport_dport_enable
 vtss_aggr_mode_t, 62
 sport_eq_dport
 vtss_ace_frame_ipv4_t, 44
 vtss_ace_frame_ipv6_t, 48
 sport_mask
 vtss_phy_ts_ip_conf_t, 373
 sport_val
 vtss_phy_ts_ip_conf_t, 372
 squelch
 vtss_phy_clock_conf_t, 314
 squelch_inv
 vtss_phy_10g_ckout_conf_t, 230
 vtss_phy_10g_sckout_conf_t, 293
 squelch_on_lopc
 vtss_phy_10g_rxckout_conf_t, 292
 squelch_on_pcs_fault
 vtss_phy_10g_rxckout_conf_t, 291
 squelsh
 vtss_synce_clock_in_t, 480
 srate
 vtss_phy_10g_pkt_gen_conf_t, 281
 src
 vtss_phy_10g_ckout_conf_t, 230
 vtss_phy_10g_sckout_conf_t, 293
 vtss_phy_clock_conf_t, 313
 src_ip
 vtss_phy_ts_fifo_sig_t, 361
 src_port_identity
 vtss_phy_ts_fifo_sig_t, 360
 sref_clk_div

vtss_phy_10g_mode_t, 265
stack_depth
 vtss_phy_ts_mpls_conf_t, 376
stack_level
 vtss_phy_ts_mpls_conf_t, 378
stack_port_a
 vtss_vstax_route_entry_t, 530
stack_port_b
 vtss_vstax_route_entry_t, 530
stack_queue
 vtss_packet_rx_queue_map_t, 201
stack_ref_point
 vtss_phy_ts_mpls_conf_t, 376
static_conf
 vtss_ewis_conf_s, 99
status
 vtss_phy_10g_status_t, 303
 vtss_phy_veriphy_result_t, 399
stripped_tag
 vtss_packet_rx_info_t, 187
stuck_at_01
 vtss_phy_10g_prbs_mon_conf_t, 290
stuck_at_par
 vtss_phy_10g_prbs_mon_conf_t, 289
sw_tstamp
 vtss_packet_rx_info_t, 189
 vtss_packet_rx_meta_t, 196
switch_frm
 vtss_packet_tx_info_t, 206
sym
 vtss_packet_tx_info_t, 210
symmetric_pause
 vtss_phy_10g_clause_37_adv_t, 232
 vtss_phy_aneg_t, 312
 vtss_port_clause_37_adv_t, 408
TRUE
 types.h, 580
table
 vtss_vstax_route_table_t, 531
tag
 vtss_ece_key_t, 79
 vtss_mirror_conf_t, 171
 vtss_packet_rx_header_t, 183
 vtss_packet_rx_info_t, 186
 vtss_packet_tx_info_t, 207
 vtss_qce_key_t, 450
 vtss_vce_key_t, 511
tag_class_enable
 vtss_qos_port_conf_t, 460
tag_default_dei
 vtss_qos_port_conf_t, 462
tag_default_pcp
 vtss_qos_port_conf_t, 462
tag_dei_map
 vtss_qos_port_conf_t, 463
tag_pcp_map
 vtss_qos_port_conf_t, 462
tag_range_mode
 vtss_phy_ts_eth_conf_t, 355
tag_remark_mode
 vtss_qos_port_conf_t, 462
tag_type
 vtss_packet_rx_info_t, 186
tagged
 vtss_ece_tag_t, 85
 vtss_qce_tag_t, 455
 vtss_vce_tag_t, 516
tagprio
 vtss_tci_t, 483
target
 vtss_inst_create_t, 146
tbi
 vtss_phy_reset_conf_t, 336
tc_op_mode
 vtss_phy_ts_init_conf_t, 370
tci
 vtss_vstax_tx_header_t, 534
tcp_ack
 vtss_ace_frame_ipv4_t, 44
 vtss_ace_frame_ipv6_t, 48
tcp_fin
 vtss_ace_frame_ipv4_t, 43
 vtss_ace_frame_ipv6_t, 47
tcp_psh
 vtss_ace_frame_ipv4_t, 44
 vtss_ace_frame_ipv6_t, 48
tcp_RST
 vtss_ace_frame_ipv4_t, 43
 vtss_ace_frame_ipv6_t, 47
tcp_SYN
 vtss_ace_frame_ipv4_t, 43
 vtss_ace_frame_ipv6_t, 47
tcp_URG
 vtss_ace_frame_ipv4_t, 44
 vtss_ace_frame_ipv6_t, 48
test_conf
 vtss_ewis_conf_s, 100
test_pattern_ana
 vtss_ewis_test_conf_s, 127
test_pattern_gen
 vtss_ewis_test_conf_s, 127
thrd_lvl_after_top
 vtss_phy_ts_mpls_conf_t, 377
thrd_lvl_before_end
 vtss_phy_ts_mpls_conf_t, 378
timeout
 vtss_mtimer_t, 172
timestamp
 vtss_phy_10g_timestamp_val_t, 305
to_cpu
 vtss_policer_ext_t, 404
tod_get_ns_cnt_cb_t
 vtss_misc_api.h, 720
top
 vtss_phy_ts_mpls_conf_t, 377
top_down

vtss_phy_ts_mpls_conf_t, 377
 topology_type
 vtss_vstax_route_table_t, 531
 tpid
 vtss_packet_port_filter_t, 178
 vtss_phy_ts_eth_conf_t, 353
 vtss_vlan_tag_t, 522
 train
 vtss_phy_10g_base_kr_status_t, 226
 training
 vtss_phy_10g_base_kr_train_aneg_t, 227
 trans_vid
 vtss_vlan_trans_grp2vlan_conf_t, 523
 trig_ch_id
 vtss_phy_10g_auto_failover_conf_t, 219
 trmthd_c0
 vtss_phy_10g_base_kr_training_t, 229
 trmthd_cm
 vtss_phy_10g_base_kr_training_t, 229
 trmthd_cp
 vtss_phy_10g_base_kr_training_t, 228
 ts
 vtss_ts_timestamp_t, 491
 ts_fifo_drop_cnt
 vtss_phy_ts_stats_t, 394
 ts_fifo_tx_cnt
 vtss_phy_ts_stats_t, 394
 ts_format
 vtss_phy_ts_ietf_mpls_ach_oam_conf_t, 367
 ts_id
 vtss_ts_id_t, 487
 ts_offset
 vtss_phy_ts_generic_action_t, 365
 ts_type
 vtss_phy_ts_generic_action_t, 365
 ts_valid
 vtss_ts_timestamp_t, 491
 tstamp_id
 vtss_packet_rx_info_t, 189
 tstamp_id_decoded
 vtss_packet_rx_info_t, 189
 tstampat_cnt
 vtss_ewis_test_status_s, 128
 tti
 vtss_ewis_tti_s, 129
 ttl
 vtss_ace_frame_ipv4_t, 41
 vtss_ace_frame_ipv6_t, 46
 vtss_vstax_port_conf_t, 529
 vtss_vstax_tx_header_t, 534
 tx_b1
 vtss_ewis_tx_passthru_s, 134
 tx_b2
 vtss_ewis_tx_passthru_s, 135
 tx_c2
 vtss_ewis_tx_oh_s, 132
 tx_ch
 vtss_phy_10g_lane_sync_conf_t, 259
 tx_clk_skew_ps
 vtss_phy_rgmii_conf_t, 338
 tx_dcc_l
 vtss_ewis_tx_oh_s, 131
 vtss_ewis_tx_passthru_s, 136
 tx_dcc_s
 vtss_ewis_tx_oh_s, 130
 vtss_ewis_tx_passthru_s, 134
 tx_discard
 vtss_evc_counters_t, 93
 tx_e1
 vtss_ewis_tx_oh_s, 130
 vtss_ewis_tx_passthru_s, 134
 tx_e2
 vtss_ewis_tx_oh_s, 131
 vtss_ewis_tx_passthru_s, 136
 tx_etherStatsBroadcastPkts
 vtss_port_rmon_counters_t, 434
 tx_etherStatsCollisions
 vtss_port_rmon_counters_t, 434
 tx_etherStatsDropEvents
 vtss_port_rmon_counters_t, 433
 tx_etherStatsMulticastPkts
 vtss_port_rmon_counters_t, 434
 tx_etherStatsOctets
 vtss_port_rmon_counters_t, 433
 tx_etherStatsPkts
 vtss_port_rmon_counters_t, 433
 tx_etherStatsPkts1024to1518Octets
 vtss_port_rmon_counters_t, 435
 tx_etherStatsPkts128to255Octets
 vtss_port_rmon_counters_t, 435
 tx_etherStatsPkts1519toMaxOctets
 vtss_port_rmon_counters_t, 435
 tx_etherStatsPkts256to511Octets
 vtss_port_rmon_counters_t, 435
 tx_etherStatsPkts512to1023Octets
 vtss_port_rmon_counters_t, 435
 tx_etherStatsPkts64Octets
 vtss_port_rmon_counters_t, 434
 tx_etherStatsPkts65to127Octets
 vtss_port_rmon_counters_t, 434
 tx_f1
 vtss_ewis_tx_oh_s, 130
 vtss_ewis_tx_passthru_s, 134
 tx_f2
 vtss_ewis_tx_oh_s, 132
 tx_fault
 vtss_sublayer_status_t, 479
 tx_fifo_hi_clk_cycs
 vtss_phy_ts_init_conf_t, 369
 tx_fifo_lo_clk_cycs
 vtss_phy_ts_init_conf_t, 370
 tx_fifo_mode
 vtss_phy_ts_init_conf_t, 369
 tx_fifo_spi_conf
 vtss_phy_ts_init_conf_t, 369
 tx_frames

vtss_basic_counters_t, 65
tx_green
 vtss_evc_counters_t, 94
tx_j0
 vtss_ewis_tx_passthru_s, 133
tx_k1
 vtss_ewis_tx_passthru_s, 135
tx_k1_k2
 vtss_ewis_tx_oh_s, 131
tx_k2
 vtss_ewis_tx_passthru_s, 135
tx_loh
 vtss_ewis_tx_passthru_s, 136
tx_macro
 vtss_phy_10g_lane_sync_conf_t, 258
tx_n1
 vtss_ewis_tx_oh_s, 132
tx_oh
 vtss_ewis_conf_s, 100
tx_oh_passthru
 vtss_ewis_conf_s, 100
tx_out_bytes
 vtss_eee_port_counter_t, 88
tx_out_bytes_get
 vtss_eee_port_counter_t, 88
tx_prio
 vtss_port_proprietary_counters_t, 428
tx_reil
 vtss_ewis_tx_passthru_s, 135
tx_remote_fault
 vtss_phy_aneg_t, 313
tx_s1
 vtss_ewis_tx_oh_s, 131
 vtss_ewis_tx_passthru_s, 136
tx_soh
 vtss_ewis_tx_passthru_s, 135
tx_ts_len
 vtss_phy_ts_init_conf_t, 369
tx_tw
 vtss_eee_port_conf_t, 86
tx_vstax_hdr
 vtss_packet_tx_info_t, 209
tx_yellow
 vtss_evc_counters_t, 94
tx_z0
 vtss_ewis_tx_oh_s, 131
 vtss_ewis_tx_passthru_s, 134
tx_z1_z2
 vtss_ewis_tx_oh_s, 132
 vtss_ewis_tx_passthru_s, 136
tx_z3_z4
 vtss_ewis_tx_oh_s, 132
type
 vtss_ace_t, 53
 vtss_dlb_policer_conf_t, 71
 vtss_ece_inner_tag_t, 77
 vtss_ece_key_t, 79
 vtss_eps_port_conf_t, 90
vtss_fan_conf_t, 138
vtss_ip_addr_t, 146
vtss_phy_10g_id_t, 250
vtss_qce_key_t, 450
vtss_routing_entry_t, 470
vtss_sfflow_port_conf_t, 474
vtss_vcap_vr_t, 503
vtss_vce_key_t, 512
types.h
 BOOL, 599
 FALSE, 580
 i16, 598
 i32, 598
 i64, 598
 i8, 597
 MAC_ADDR_BROADCAST, 589
 PRI64, 579
 PRIu64, 578
 PRIx64, 579
 TRUE, 580
 u16, 598
 u32, 599
 u64, 599
 u8, 598
 uintptr_t, 599
 VTSS_ACL_POLICER_NO_END, 593
 VTSS_ACL_POLICER_NO_START, 593
 VTSS_ACL_POLICERS, 593
 VTSS_ACL_POLICIES, 594
 VTSS_ACL_POLICY_NO_END, 594
 VTSS_ACL_POLICY_NO_MAX, 594
 VTSS_ACL_POLICY_NO_MIN, 594
 VTSS_ACL_POLICY_NO_NONE, 593
 VTSS_ACL_POLICY_NO_START, 594
 VTSS_AGGR_NO_END, 590
 VTSS_AGGR_NO_NONE, 589
 VTSS_AGGR_NO_START, 590
 VTSS_AGGRS, 589
 VTSS_BIT64, 579
 VTSS_BITMASK64, 579
 VTSS_BITRATE_DISABLED, 587
 VTSS_CLOCK_IDENTITY_LENGTH, 597
 VTSS_DEI_ARRAY_SIZE, 586
 VTSS_DEI_END, 586
 VTSS_DEI_START, 585
 VTSS_DEIS, 585
 VTSS_DPL_ARRAY_SIZE, 587
 VTSS_DPL_END, 587
 VTSS_DPL_START, 586
 VTSS_DPLS, 586
 VTSS_ENCODE_BITFIELD64, 580
 VTSS_ENCODE_BITMASK64, 580
 VTSS_ETYPE_VTSS, 588
 VTSS_EVCS, 589
 VTSS_EXTRACT_BITFIELD64, 579
 VTSS_GLAG_NO_END, 591
 VTSS_GLAG_NO_NONE, 590
 VTSS_GLAG_NO_START, 590

VTSS_GLAG_PORT_ARRAY_SIZE, 591
 VTSS_GLAG_PORT_END, 591
 VTSS_GLAG_PORT_START, 591
 VTSS_GLAG_PORTS, 591
 VTSS_GLAGS, 590
 VTSS_HQOS_COUNT, 595
 VTSS_HQOS_ID_NONE, 595
 VTSS_INTERVAL_MS, 596
 VTSS_INTERVAL_NS, 596
 VTSS_INTERVAL_PS, 596
 VTSS_INTERVAL_SEC, 596
 VTSS_INTERVAL_US, 596
 VTSS_ISDX_NONE, 589
 VTSS_MAC_ADDR_SZ_BYTES, 588
 VTSS_MAX_TIMEINTERVAL, 595
 VTSS_ONE_MILL, 595
 VTSS_ONE_MIA, 595
 VTSS_PACKET_RATE_DISABLED, 581
 VTSS_PACKET_RX_GRP_CNT, 592
 VTSS_PACKET_RX_QUEUE_CNT, 592
 VTSS_PACKET_RX_QUEUE_END, 593
 VTSS_PACKET_RX_QUEUE_NONE, 592
 VTSS_PACKET_RX_QUEUE_START, 592
 VTSS_PACKET_TX_GRP_CNT, 592
 VTSS_PCP_ARRAY_SIZE, 585
 VTSS_PCP_END, 585
 VTSS_PCP_START, 585
 VTSS_PCPS, 584
 VTSS_PORT_ARRAY_SIZE, 582
 VTSS_PORT_COUNT, 581
 VTSS_PORT_IS_PORT, 582
 VTSS_PORT_NO_CPU, 582
 VTSS_PORT_NO_END, 582
 VTSS_PORT_NO_NONE, 581
 VTSS_PORT_NO_START, 582
 VTSS_PORTS, 581
 VTSS_PRIO_ARRAY_SIZE, 583
 VTSS_PRIO_END, 583
 VTSS_PRIO_NO_NONE, 583
 VTSS_PRIO_START, 583
 VTSS_PRIOS, 583
 VTSS_QUEUE_ARRAY_SIZE, 584
 VTSS_QUEUE_END, 584
 VTSS_QUEUE_START, 584
 VTSS_QUEUES, 584
 VTSS_SEC_NS_INTERVAL, 597
 VTSS_SYNCE_CLK_PORT_ARRAY_SIZE, 597
 VTSS_VID_ALL, 588
 VTSS_VID_DEFAULT, 587
 VTSS_VID_NULL, 587
 VTSS_VID_RESERVED, 588
 VTSS_VIDS, 588
 vtss_ece_dir_t, 607
 vtss_ece_inner_tag_type_t, 608
 vtss_ece_pop_tag_t, 608
 vtss_hqos_sch_mode_t, 609
 vtss_ip_type_t, 606
 vtss_isdx_t, 600
 vtss_mac_addr_t, 599
 vtss_mem_flags_t, 603
 vtss_packet_reg_type_t, 605
 vtss_packet_rx_grp_t, 600
 vtss_packet_tx_grp_t, 600
 vtss_policer_type_t, 605
 vtss_port_interface_t, 603
 vtss_serdes_mode_t, 604
 vtss_storm_policer_mode_t, 605
 vtss_vcap_bit_t, 606
 vtss_vcap_key_type_t, 607
 vtss_vcap_vr_type_t, 607
 vtss_vdd_t, 606
 vtss_vlan_frame_t, 605

u16
 types.h, 598
 u32
 types.h, 599
 u64
 types.h, 599
 u8
 types.h, 598
 UNSIGNED_STORAGE_COUNT
 vtss_phy_10g_api.h, 808
 uc_no_flood
 vtss_policer_ext_t, 404
 uint
 vtss_os_custom.h, 748
 uintptr_t
 types.h, 599
 ulong
 vtss_os_custom.h, 748
 uncorrected_block_cnt
 vtss_phy_10g_kr_status_fec_t, 256
 unicast
 vtss_policer_ext_t, 403
 unidir
 vtss_phy_conf_t, 317
 unidirectional_ability
 vtss_port_status_t, 440
 unknown
 vtss_ace_frame_arp_t, 37
 untagged_vid
 vtss_vlan_port_conf_t, 520
 update
 vtss_phy_10g_pkt_mon_conf_t, 282
 upper
 vtss_phy_ts_eth_conf_t, 355
 vtss_phy_ts_mpls_l1_lrnge_t, 379
 vtss_phy_ts_ptp_conf_t, 387
 vtss_phy_ts_y1731_oam_conf_t, 396
 upsid
 vtss_mac_table_entry_t, 168
 vtss_vstax_glag_entry_t, 528
 vtss_vstax_rx_header_t, 532
 vtss_vstax_tx_header_t, 534
 upsid_0
 vtss_vstax_conf_t, 526

upsid_1
 vtss_vstax_conf_t, 527

upspn
 vtss_mac_table_entry_t, 168
 vtss_vstax_glag_entry_t, 528

use_as_intrpt
 vtss_gpio_10g_gpio_mode_t, 141

use_conf
 vtss_phy_10g_mode_t, 266

usr_prio
 vtss_ace_vlan_t, 56
 vtss_qos_port_conf_t, 460

v
 vtss_vcap_vr_t, 503

v3
 vtss_phy_10g_ob_status_t, 275

v4
 vtss_phy_10g_ob_status_t, 275

v5
 vtss_phy_10g_ob_status_t, 276

v_gpio
 vtss_phy_10g_auto_failover_conf_t, 219

VIPER_B_FIFO_RESET
 options.h, 553

VTSS_10G_PHY_GPIO_MAX
 vtss_phy_10g_api.h, 809

VTSS_10G_PHY_GPIO_MAX
 vtss_phy_10g_api.h, 809

VTSS_ACE_ID_LAST
 vtss_security_api.h, 1059

VTSS_ACL_POLICER_NO_END
 types.h, 593

VTSS_ACL_POLICER_NO_START
 types.h, 593

VTSS_ACL_POLICERS
 types.h, 593

VTSS_ACL_POLICIES
 types.h, 594

VTSS_ACL_POLICY_NO_END
 types.h, 594

VTSS_ACL_POLICY_NO_MAX
 types.h, 594

VTSS_ACL_POLICY_NO_MIN
 types.h, 594

VTSS_ACL_POLICY_NO_NONE
 types.h, 593

VTSS_ACL_POLICY_NO_START
 types.h, 594

VTSS_AGGR_NO_END
 types.h, 590

VTSS_AGGR_NO_NONE
 types.h, 589

VTSS_AGGR_NO_START
 types.h, 590

VTSS_AGGRS
 types.h, 589

VTSS_ARCH_JAGUAR_1
 options.h, 540

VTSS_ARCH_JAGUAR_1_CE_SWITCH
 options.h, 540

VTSS_ARCH_MALIBU_B
 options.h, 556

VTSS_ARCH_MALIBU
 options.h, 555

VTSS_ARCH_VENICE_C
 options.h, 556

VTSS_ARP_CNT
 vtss_l3_api.h, 705

VTSS_ARP_IPV4_RELATIONS
 vtss_l3_api.h, 706

VTSS_ARP_IPV6_RELATIONS
 vtss_l3_api.h, 706

VTSS_BIT64
 types.h, 579

VTSS_BITMASK64
 types.h, 579

VTSS_BITRATE_DISABLED
 types.h, 587

VTSS_CHIP_10G_PHY
 options.h, 542

VTSS_CHIP CU PHY
 options.h, 554

VTSS_CLOCK_IDENTITY_LENGTH
 types.h, 597

VTSS_DEI_ARRAY_SIZE
 types.h, 586

VTSS_DEI_END
 types.h, 586

VTSS_DEI_START
 types.h, 585

VTSS_DEIS
 types.h, 585

VTSS_DIV64
 vtss_os_custom.h, 749
 vtss_os_ecos.h, 754
 vtss_os_linux.h, 766

VTSS_DPL_ARRAY_SIZE
 types.h, 587

VTSS_DPL_END
 types.h, 587

VTSS_DPL_START
 types.h, 586

VTSS_DPLS
 types.h, 586

VTSS_ECE_ID_LAST
 vtss_evc_api.h, 613

VTSS_ENCODE_BITFIELD64
 types.h, 580

VTSS_ENCODE_BITMASK64
 types.h, 580

VTSS_ERPI_ARRAY_SIZE
 vtss_l2_api.h, 653

VTSS_ERPI_END
 vtss_l2_api.h, 653

VTSS_ERPI_START
 vtss_l2_api.h, 653

VTSS_ERPIS
 vtss_l2_api.h, 652

VTSSETYPE_VTSS
 types.h, 588

VTSS_EVC_ID_NONE
 vtss_evc_api.h, 613

VTSS_EVC_POLICER_ID_DISCARD
 vtss_evc_api.h, 613

VTSS_EVC_POLICER_ID_EVC
 vtss_evc_api.h, 613

VTSS_EVC_POLICER_ID_NONE
 vtss_evc_api.h, 613

VTSS_EVC_POLICERS
 vtss_evc_api.h, 612

VTSS_EVCS
 types.h, 589

VTSS_EWIS_AISL_EV
 vtss_wis_api.h, 1094

VTSS_EWIS_AISP_EV
 vtss_wis_api.h, 1095

VTSS_EWIS_B1_NZ_EV
 vtss_wis_api.h, 1098

VTSS_EWIS_B1_THRESH_EV
 vtss_wis_api.h, 1099

VTSS_EWIS_B2_NZ_EV
 vtss_wis_api.h, 1098

VTSS_EWIS_B2_THRESH_EV
 vtss_wis_api.h, 1099

VTSS_EWIS_B3_NZ_EV
 vtss_wis_api.h, 1098

VTSS_EWIS_B3_THRESH_EV
 vtss_wis_api.h, 1099

VTSS_EWIS_FAIS_EV
 vtss_wis_api.h, 1094

VTSS_EWIS_FERDIP_EV
 vtss_wis_api.h, 1097

VTSS_EWIS_FEUNEQP_EV
 vtss_wis_api.h, 1096

VTSS_EWIS_FPLM_EV
 vtss_wis_api.h, 1093

VTSS_EWIS_HIGH_BER_EV
 vtss_wis_api.h, 1097

VTSS_EWIS_LCDP_EV
 vtss_wis_api.h, 1095

VTSS_EWIS_LOF_EV
 vtss_wis_api.h, 1094

VTSS_EWIS_LOPC_EV
 vtss_wis_api.h, 1096

VTSS_EWIS_LOPP_EV
 vtss_wis_api.h, 1095

VTSS_EWIS_LOS_EV
 vtss_wis_api.h, 1094

VTSS_EWIS_MODULE_EV
 vtss_wis_api.h, 1095

VTSS_EWIS_PCS_RECEIVE_FAULT_PEND
 vtss_wis_api.h, 1097

VTSS_EWIS_PLMP_EV
 vtss_wis_api.h, 1095

VTSS_EWIS_RDIL_EV
 vtss_wis_api.h, 1094

VTSS_EWIS_REIL_EV
 vtss_wis_api.h, 1097

VTSS_EWIS_REIL_NZ_EV
 vtss_wis_api.h, 1098

VTSS_EWIS_REIL_THRESH_EV
 vtss_wis_api.h, 1099

VTSS_EWIS_REIP_EV
 vtss_wis_api.h, 1097

VTSS_EWIS_REIP_NZ_EV
 vtss_wis_api.h, 1098

VTSS_EWIS_REIP_THRESH_EV
 vtss_wis_api.h, 1099

VTSS_EWIS_RXLOL_EV
 vtss_wis_api.h, 1096

VTSS_EWIS_SEF_EV
 vtss_wis_api.h, 1093

VTSS_EWIS_TXLOL_EV
 vtss_wis_api.h, 1096

VTSS_EWIS_UNEQP_EV
 vtss_wis_api.h, 1096

VTSS_EXTRACT_BITFIELD64
 types.h, 579

VTSS_FEATURE_10GBASE_KR
 options.h, 552

VTSS_FEATURE_10G
 options.h, 543

VTSS_FEATURE_ACL_V1
 options.h, 551

VTSS_FEATURE_ACL
 options.h, 551

VTSS_FEATURE_AGGR_GLAG
 options.h, 541

VTSS_FEATURE_CLAUSE_37
 options.h, 543

VTSS_FEATURE_E_TREE
 options.h, 540

VTSS_FEATURE_EDC_FW_LOAD
 options.h, 555

VTSS_FEATURE_EEE
 options.h, 552

VTSS_FEATURE_EVC
 options.h, 540

VTSS_FEATURE_EXC_COL_CONT
 options.h, 544

VTSS_FEATURE_FAN
 options.h, 541, 551

VTSS_FEATURE_IRQ_CONTROL
 options.h, 552

VTSS_FEATURE_LAYER2
 options.h, 549

VTSS_FEATURE_LAYER3
 options.h, 550

VTSS_FEATURE_LED_POW_REDUC
 options.h, 552

VTSS_FEATURE_MAC_AGE_AUTO
 options.h, 550

VTSS_FEATURE_MAC_CPU_QUEUE
 options.h, 550

VTSS_FEATURE_MISC
 options.h, 542

VTSS_FEATURE_NPI
 options.h, 543

VTSS_FEATURE_PACKET_GROUPING
 options.h, 548

VTSS_FEATURE_PACKET_PORT_REG
 options.h, 549

VTSS_FEATURE_PACKET_RX
 options.h, 548

VTSS_FEATURE_PACKET_TX
 options.h, 548

VTSS_FEATURE_PACKET
 options.h, 548

VTSS_FEATURE_PHY_TIMESTAMP
 options.h, 540

VTSS_FEATURE_PORT_CNT_BRIDGE
 options.h, 544

VTSS_FEATURE_PORT_CNT_ETHER_LIKE
 options.h, 544

VTSS_FEATURE_PORT_CONTROL
 options.h, 543

VTSS_FEATURE_PORT_IFH
 options.h, 543

VTSS_FEATURE_PORT_MUX
 options.h, 550

VTSS_FEATURE_PVLAN
 options.h, 542

VTSS_FEATURE_QCL_V2
 options.h, 545

VTSS_FEATURE_QCL
 options.h, 544

VTSS_FEATURE_QOS_CLASSIFICATION_V2
 options.h, 547

VTSS_FEATURE_QOS_DSCP_REMARK_V2
 options.h, 547

VTSS_FEATURE_QOS_DSCP_REMARK
 options.h, 547

VTSS_FEATURE_QOS_EGRESS_QUEUE_SHAPE_RS_EB
 options.h, 547

VTSS_FEATURE_QOS_EGRESS_QUEUE_SHAPE_RS
 options.h, 547

VTSS_FEATURE_QOS_POLICER_DLDB
 options.h, 553

VTSS_FEATURE_QOS_PORT_POLICER_EXT_DPBL
 options.h, 545

VTSS_FEATURE_QOS_PORT_POLICER_EXT_FPS
 options.h, 545

VTSS_FEATURE_QOS_PORT_POLICER_EXT_FC
 options.h, 545

VTSS_FEATURE_QOS_PORT_POLICER_EXT_TTM
 options.h, 546

VTSS_FEATURE_QOS_PORT_POLICER_EXT
 options.h, 545

VTSS_FEATURE_QOS_QUEUE_POLICER
 options.h, 546

VTSS_FEATURE_QOS_QUEUE_TX
 options.h, 546

VTSS_FEATURE_QOS_SCHEDULER_V2
 options.h, 546

VTSS_FEATURE_QOS_TAG_REMARK_V2
 options.h, 546

VTSS_FEATURE_QOS_WRED
 options.h, 548

VTSS_FEATURE_QOS
 options.h, 544

VTSS_FEATURE_SERDES_MACRO_SETTINGS
 options.h, 552

VTSS_FEATURE_SERIAL_GPIO
 options.h, 542

VTSS_FEATURE_SFLOW
 options.h, 553

VTSS_FEATURE_SYNCE_10G
 options.h, 555

VTSS_FEATURE_SYNCE
 options.h, 551

VTSS_FEATURE_TIMESTAMP
 options.h, 540

VTSS_FEATURE_VCAP
 options.h, 550, 556

VTSS_FEATURE_VCL
 options.h, 551

VTSS_FEATURE_VLAN_PORT_V2
 options.h, 549

VTSS_FEATURE_VLAN_SVL
 options.h, 549

VTSS_FEATURE_VLAN_TRANSLATION
 options.h, 553

VTSS_FEATURE_VLAN_TX_TAG
 options.h, 549

VTSS_FEATURE_VSTAX_V2
 options.h, 541

VTSS_FEATURE_VSTAX
 options.h, 541

VTSS_FEATURE_WARM_START
 options.h, 554, 555

VTSS_FEATURE_WIS
 options.h, 555

VTSS_FRAME_GAP_DEFAULT
 vtss_port_api.h, 1034

VTSS_GLAG_NO_END
 types.h, 591

VTSS_GLAG_NO_NONE
 types.h, 590

VTSS_GLAG_NO_START
 types.h, 590

VTSS_GLAG_PORT_ARRAY_SIZE
 types.h, 591

VTSS_GLAG_PORT_END
 types.h, 591

VTSS_GLAG_PORT_START
 types.h, 591

VTSS_GLAG_PORTS
 types.h, 591

VTSS_GLAGS
 types.h, 590

VTSS_HQOS_COUNT
 types.h, 595

VTSS_HQOS_ID_NONE
 types.h, 595

VTSS_I2C_NO_MULTIPLEXER
 vtss_init_api.h, 625

VTSS_INTERVAL_MS
 types.h, 596

VTSS_INTERVAL_NS
 types.h, 596

VTSS_INTERVAL_PS
 types.h, 596

VTSS_INTERVAL_SEC
 types.h, 596

VTSS_INTERVAL_US
 types.h, 596

VTSS_ISDX_NONE
 types.h, 589

VTSS_JR1_ARP_CNT
 vtss_l3_api.h, 705

VTSS_JR1_LPM_CNT
 vtss_l3_api.h, 704

VTSS_JR1_PACKET_HDR_SIZE_BYTES
 vtss_packet_api.h, 773

VTSS_JR1_RLEG_CNT
 vtss_l3_api.h, 705

VTSS_JR1_RX_IFH_SIZE
 vtss_packet_api.h, 774

VTSS_JR2_PACKET_HDR_SIZE_BYTES
 vtss_packet_api.h, 773

VTSS_JR2_RX_IFH_SIZE
 vtss_packet_api.h, 775

VTSS_L26_PACKET_HDR_SIZE_BYTES
 vtss_packet_api.h, 774

VTSS_L26_RX_IFH_SIZE
 vtss_packet_api.h, 775

VTSS_LABS
 vtss_os_custom.h, 750
 vtss_os_ecos.h, 755
 vtss_os_linux.h, 766

VTSS_LLABS
 vtss_os_custom.h, 750
 vtss_os_ecos.h, 755
 vtss_os_linux.h, 767

VTSS_LPM_CNT
 vtss_l3_api.h, 705

VTSS_MAC_ADDR_SZ_BYTES
 types.h, 588

VTSS_MAC_ADDRS
 vtss_l2_api.h, 646

VTSS_MAX_FRAME_LENGTH_MAX
 vtss_port_api.h, 1035

VTSS_MAX_FRAME_LENGTH_STANDARD
 vtss_port_api.h, 1034

VTSS_MAX_TIMEINTERVAL
 types.h, 595

VTSS_MOD64
 vtss_os_custom.h, 749
 vtss_os_ecos.h, 755
 vtss_os_linux.h, 766

VTSS_MSLEEP
 vtss_os_custom.h, 748
 vtss_os_ecos.h, 753
 vtss_os_linux.h, 764

VTSS_MSTI_ARRAY_SIZE
 vtss_l2_api.h, 648

VTSS_MSTI_END
 vtss_l2_api.h, 648

VTSS_MSTI_START
 vtss_l2_api.h, 647

VTSS_MSTIS
 vtss_l2_api.h, 647

VTSS_MTIMER_CANCEL
 vtss_os_custom.h, 749
 vtss_os_ecos.h, 754
 vtss_os_linux.h, 765

VTSS_MTIMER_START
 vtss_os_custom.h, 749
 vtss_os_ecos.h, 754
 vtss_os_linux.h, 764

VTSS_MTIMER_TIMEOUT
 vtss_os_custom.h, 749
 vtss_os_ecos.h, 754
 vtss_os_linux.h, 764

VTSS_NSLEEP
 vtss_os_ecos.h, 753
 vtss_os_linux.h, 763

VTSS_ONE_MILL
 types.h, 595

VTSS_ONE_MIA
 types.h, 595

VTSS_OPT_TRACE
 options.h, 554

VTSS_OPT_TS_SPI_FPGA
 options.h, 542

VTSS_OPT_VAUI_EQ_CTRL
 options.h, 554

VTSS_OS_BIG_ENDIAN
 vtss_os_ecos.h, 758
 vtss_os_linux.h, 763

VTSS_OS_COMPILER_ATTRIBUTE_ALIGNED
 vtss_os_ecos.h, 757

VTSS_OS_CTZ64
 vtss_os_custom.h, 750
 vtss_os_ecos.h, 756
 vtss_os_linux.h, 767

VTSS_OS_CTZ
 vtss_os_custom.h, 750
 vtss_os_ecos.h, 755
 vtss_os_linux.h, 767

VTSS_OS_DCACHE_FLUSH
 vtss_os_ecos.h, 758

VTSS_OS_DCACHE_INVALIDATE
vtss_os_ecos.h, 758

VTSS_OS_DCACHE_LINE_SIZE_BYTES
vtss_os_ecos.h, 757

VTSS_OS_FREE
vtss_os_custom.h, 751

vtss_os_ecos.h, 756

vtss_os_linux.h, 768

VTSS_OS_INTERRUPT_DISABLE
vtss_os_ecos.h, 760

VTSS_OS_INTERRUPT_FLAGS
vtss_os_ecos.h, 760

VTSS_OS_INTERRUPT_RESTORE
vtss_os_ecos.h, 760

VTSS_OS_MALLOC
vtss_os_custom.h, 751

vtss_os_ecos.h, 756

vtss_os_linux.h, 768

VTSS_OS_NTOHL
vtss_os_ecos.h, 759

vtss_os_linux.h, 763

VTSS_OS_RAND
vtss_os_custom.h, 751

vtss_os_ecos.h, 757

vtss_os_linux.h, 768

VTSS_OS_reordered_BARRIER
vtss_os_ecos.h, 757

VTSS_OS_SCHEDULER_FLAGS
vtss_os_ecos.h, 759

vtss_os_linux.h, 765

VTSS_OS_SCHEDULER_LOCK
vtss_os_ecos.h, 759

vtss_os_linux.h, 765

VTSS_OS_SCHEDULER_UNLOCK
vtss_os_ecos.h, 759

vtss_os_linux.h, 766

VTSS_OS_TIMESTAMP_TYPE
vtss_misc_api.h, 720

VTSS_OS_TIMESTAMP
vtss_misc_api.h, 720

VTSS_OS_VIRT_TO_PHYS
vtss_os_ecos.h, 758

VTSS_PACKET_HDR_SIZE_BYTES
vtss_packet_api.h, 774

VTSS_PACKET_RATE_DISABLED
types.h, 581

VTSS_PACKET_RX_GRP_CNT
types.h, 592

VTSS_PACKET_RX_QUEUE_CNT
types.h, 592

VTSS_PACKET_RX_QUEUE_END
types.h, 593

VTSS_PACKET_RX_QUEUE_NONE
types.h, 592

VTSS_PACKET_RX_QUEUE_START
types.h, 592

VTSS_PACKET_TX_GRP_CNT
types.h, 592

VTSS_PACKET_TX_IFH_MAX
vtss_packet_api.h, 775

VTSS_PCP_ARRAY_SIZE
types.h, 585

VTSS_PCP_END
types.h, 585

VTSS_PCP_START
types.h, 585

VTSS_PCPS
types.h, 584

VTSS_PHY_10G_FEC_FIXED_CNT_THRESH_EV
vtss_phy_10g_api.h, 818

VTSS_PHY_10G_FEC_UNFIXED_CNT_THRESH_EV
vtss_phy_10g_api.h, 818

VTSS_PHY_10G_FIFO_SYNC
options.h, 553

VTSS_PHY_10G_GPIO_INT_AGG0_EV
vtss_phy_10g_api.h, 819

VTSS_PHY_10G_GPIO_INT_AGG1_EV
vtss_phy_10g_api.h, 819

VTSS_PHY_10G_GPIO_INT_AGG2_EV
vtss_phy_10g_api.h, 819

VTSS_PHY_10G_GPIO_INT_AGG3_EV
vtss_phy_10g_api.h, 819

VTSS_PHY_10G_HIGH_BER_EV
vtss_phy_10g_api.h, 810

VTSS_PHY_10G_HIGHBER_EV
vtss_phy_10g_api.h, 818

VTSS_PHY_10G_HOST_MAC_LOCAL_FAULT_EV
vtss_phy_10g_api.h, 820

VTSS_PHY_10G_HOST_MAC_REMOTE_FAULT_EV
vtss_phy_10g_api.h, 821

VTSS_PHY_10G_LINE_MAC_LOCAL_FAULT_EV
vtss_phy_10g_api.h, 820

VTSS_PHY_10G_LINE_MAC_REMOTE_FAULT_EV
vtss_phy_10g_api.h, 820

VTSS_PHY_10G_LINK_LOS_EV
vtss_phy_10g_api.h, 810

VTSS_PHY_10G_LOPC_EV
vtss_phy_10g_api.h, 810

VTSS_PHY_10G_MACSEC_DISABLED
vtss_phy_10g_api.h, 809

VTSS_PHY_10G_MACSEC_KEY_128
vtss_phy_10g_api.h, 809

VTSS_PHY_10G_MODULE_STAT_EV
vtss_phy_10g_api.h, 811

VTSS_PHY_10G_ONE_LINE_ACTIVE
vtss_phy_10g_api.h, 808

VTSS_PHY_10G_PCS_RECEIVE_FAULT_EV
vtss_phy_10g_api.h, 811

VTSS_PHY_10G_RX_BLK_DEC_CNT_THRESH_EV
vtss_phy_10g_api.h, 817

VTSS_PHY_10G_RX_CHAR_DEC_CNT_THRESH_EV
vtss_phy_10g_api.h, 817

VTSS_PHY_10G_RX_LINK_STAT_EV
vtss_phy_10g_api.h, 818, 819

VTSS_PHY_10G_RX_LOL_EV

vtss_phy_10g_api.h, [810](#), [816](#)
VTSS_PHY_10G_RX_LOS_EV
 vtss_phy_10g_api.h, [816](#)
VTSS_PHY_10G_RX_SEQ_CNT_THRESH_EV
 vtss_phy_10g_api.h, [817](#)
VTSS_PHY_10G_TIMESTAMP_DISABLED
 vtss_phy_10g_api.h, [809](#)
VTSS_PHY_10G_TX_BLK_ENC_CNT_THRESH_EV
 vtss_phy_10g_api.h, [817](#)
VTSS_PHY_10G_TX_CHAR_ENC_CNT_THRESH_EV
 vtss_phy_10g_api.h, [817](#)
VTSS_PHY_10G_TX_LOL_EV
 vtss_phy_10g_api.h, [810](#), [816](#)
VTSS_PHY_10G_TX_SEQ_CNT_THRESH_EV
 vtss_phy_10g_api.h, [818](#)
VTSS_PHY_1G_HOST_AUTONEG_RESTART_EV
 vtss_phy_10g_api.h, [820](#)
VTSS_PHY_1G_LINE_AUTONEG_RESTART_EV
 vtss_phy_10g_api.h, [820](#)
VTSS_PHY_ACTIPHY_PWR
 vtss_phy_api.h, [897](#)
VTSS_PHY_EWIS_AISL_EV
 vtss_phy_10g_api.h, [812](#)
VTSS_PHY_EWIS_AISP_EV
 vtss_phy_10g_api.h, [813](#)
VTSS_PHY_EWIS_B1_NZ_EV
 vtss_phy_10g_api.h, [814](#)
VTSS_PHY_EWIS_B1_THRESH_EV
 vtss_phy_10g_api.h, [815](#)
VTSS_PHY_EWIS_B2_NZ_EV
 vtss_phy_10g_api.h, [814](#)
VTSS_PHY_EWIS_B2_THRESH_EV
 vtss_phy_10g_api.h, [815](#)
VTSS_PHY_EWIS_B3_NZ_EV
 vtss_phy_10g_api.h, [814](#)
VTSS_PHY_EWIS_B3_THRESH_EV
 vtss_phy_10g_api.h, [815](#)
VTSS_PHY_EWIS_FAIS_EV
 vtss_phy_10g_api.h, [811](#)
VTSS_PHY_EWIS_FERDIP_EV
 vtss_phy_10g_api.h, [813](#)
VTSS_PHY_EWIS_FEUNEQP_EV
 vtss_phy_10g_api.h, [813](#)
VTSS_PHY_EWIS_FPLM_EV
 vtss_phy_10g_api.h, [811](#)
VTSS_PHY_EWIS_LCDP_EV
 vtss_phy_10g_api.h, [812](#)
VTSS_PHY_EWIS_LOF_EV
 vtss_phy_10g_api.h, [812](#)
VTSS_PHY_EWIS_LOPP_EV
 vtss_phy_10g_api.h, [813](#)
VTSS_PHY_EWIS_PLMP_EV
 vtss_phy_10g_api.h, [812](#)
VTSS_PHY_EWIS_RDIL_EV
 vtss_phy_10g_api.h, [812](#)
VTSS_PHY_EWIS_REIL_EV
 vtss_phy_10g_api.h, [814](#)
VTSS_PHY_EWIS_REIL_NZ_EV
 vtss_phy_10g_api.h, [815](#)
VTSS_PHY_EWIS_REL_THRESH_EV
 vtss_phy_10g_api.h, [816](#)
VTSS_PHY_EWIS_REIP_EV
 vtss_phy_10g_api.h, [814](#)
VTSS_PHY_EWIS_REIP_NZ_EV
 vtss_phy_10g_api.h, [815](#)
VTSS_PHY_EWIS_REIP_THRESH_EV
 vtss_phy_10g_api.h, [816](#)
VTSS_PHY_EWIS_SEF_EV
 vtss_phy_10g_api.h, [811](#)
VTSS_PHY_EWIS_UNEQP_EV
 vtss_phy_10g_api.h, [813](#)
VTSS_PHY_LINK_AMS_EV
 vtss_phy_api.h, [901](#)
VTSS_PHY_LINK_AUTO_NEG_COMPLETE_EV
 vtss_phy_api.h, [902](#)
VTSS_PHY_LINK_AUTO_NEG_ERROR_EV
 vtss_phy_api.h, [902](#)
VTSS_PHY_LINK_DOWN_PWR
 vtss_phy_api.h, [897](#)
VTSS_PHY_LINK_EXT_EEE_LINKFAIL_EV
 vtss_phy_api.h, [905](#)
VTSS_PHY_LINK_EXT_EEE_WAIT_RX_TQ_EV
 vtss_phy_api.h, [905](#)
VTSS_PHY_LINK_EXT_EEE_WAIT_TS_EV
 vtss_phy_api.h, [905](#)
VTSS_PHY_LINK_EXT_EEE_WAKE_ERR_EV
 vtss_phy_api.h, [904](#)
VTSS_PHY_LINK_EXT_MACSEC_EGRESS_EV
 vtss_phy_api.h, [906](#)
VTSS_PHY_LINK_EXT_MACSEC_FC_BUFF_EV
 vtss_phy_api.h, [906](#)
VTSS_PHY_LINK_EXT_MACSEC_HOST_MAC_EV
 vtss_phy_api.h, [905](#)
VTSS_PHY_LINK_EXT_MACSEC_INGRESS_EV
 vtss_phy_api.h, [906](#)
VTSS_PHY_LINK_EXT_MACSEC_LINE_MAC_EV
 vtss_phy_api.h, [906](#)
VTSS_PHY_LINK_EXT_MEM_INT_RING_EV
 vtss_phy_api.h, [906](#)
VTSS_PHY_LINK_EXT_RR_SW_COMPL_EV
 vtss_phy_api.h, [905](#)
VTSS_PHY_LINK_EXTENDED_REG_INT_EV
 vtss_phy_api.h, [904](#)
VTSS_PHY_LINK_FALSE_CARRIER_INT_EV
 vtss_phy_api.h, [903](#)
VTSS_PHY_LINK_FDX_STATE_CHANGE_EV
 vtss_phy_api.h, [902](#)
VTSS_PHY_LINK_FFAIL_EV
 vtss_phy_api.h, [901](#)
VTSS_PHY_LINK_INLINE_POW_DEV_DETECT_EV
 vtss_phy_api.h, [902](#)
VTSS_PHY_LINK_LINK_SPEED_DS_DETECT_EV
 vtss_phy_api.h, [903](#)
VTSS_PHY_LINK_LOS_EV
 vtss_phy_api.h, [901](#)
VTSS_PHY_LINK_MASTER_SLAVE_RES_ERR_EV

vtss_phy_api.h, 904
VTSS_PHY_LINK_RX_ER_INT_EV
vtss_phy_api.h, 904
VTSS_PHY_LINK_RX_FIFO_OVERFLOW_INT_EV
vtss_phy_api.h, 903
VTSS_PHY_LINK_SPEED_STATE_CHANGE_EV
vtss_phy_api.h, 902
VTSS_PHY_LINK_SYMBOL_ERR_INT_EV
vtss_phy_api.h, 903
VTSS_PHY_LINK_TX_FIFO_OVERFLOW_INT_EV
vtss_phy_api.h, 903
VTSS_PHY_LINK_UP_FULL_PWR
vtss_phy_api.h, 897
VTSS_PHY_LINK_WAKE_ON_LAN_INT_EV
vtss_phy_api.h, 904
VTSS_PHY_OPT_VERIPHYS
options.h, 554
VTSS_PHY_PAGE_0x2DAF
vtss_phy_api.h, 900
VTSS_PHY_PAGE_1588
vtss_phy_api.h, 899
VTSS_PHY_PAGE_EXTENDED_2
vtss_phy_api.h, 898
VTSS_PHY_PAGE_EXTENDED_3
vtss_phy_api.h, 898
VTSS_PHY_PAGE_EXTENDED_4
vtss_phy_api.h, 899
VTSS_PHY_PAGE_EXTENDED
vtss_phy_api.h, 898
VTSS_PHY_PAGE_GPIO
vtss_phy_api.h, 899
VTSS_PHY_PAGE_MACSEC
vtss_phy_api.h, 899
VTSS_PHY_PAGE_STANDARD
vtss_phy_api.h, 898
VTSS_PHY_PAGE_TEST
vtss_phy_api.h, 899
VTSS_PHY_PAGE_TR
vtss_phy_api.h, 900
VTSS_PHY_POWER_ACTIPHY_BIT
phy.h, 557
vtss_phy_api.h, 896
VTSS_PHY_POWER_DYNAMIC_BIT
phy.h, 557
vtss_phy_api.h, 896
VTSS_PHY_RECov_CLK1
vtss_phy_api.h, 897
VTSS_PHY_RECov_CLK2
vtss_phy_api.h, 897
VTSS_PHY_RECov_CLK_NUM
vtss_phy_api.h, 898
VTSS_PHY_REG_EXTENDED
vtss_phy_api.h, 900
VTSS_PHY_REG_GPIO
vtss_phy_api.h, 900
VTSS_PHY_REG_STANDARD
vtss_phy_api.h, 900
VTSS_PHY_REG_TEST

vtss_phy_api.h, 901
VTSS_PHY_REG_TR
vtss_phy_api.h, 901
VTSS_PHY_TS_8487_XAUI_SEL_0
vtss_phy_ts_api.h, 982
VTSS_PHY_TS_8487_XAUI_SEL_1
vtss_phy_ts_api.h, 982
VTSS_PHY_TS_DATA_IN_RSRVD_FIELD
vtss_phy_ts_api.h, 981
VTSS_PHY_TS_EGR_DATAPATH_RESET
vtss_phy_ts_api.h, 983
VTSS_PHY_TS_EGR_ENGINE_ERR
vtss_phy_ts_api.h, 981
VTSS_PHY_TS_EGR_FIFO_OVERFLOW
vtss_phy_ts_api.h, 981
VTSS_PHY_TS_EGR_FIFO_RESET
vtss_phy_ts_api.h, 983
VTSS_PHY_TS_EGR_LTC2_RESET
vtss_phy_ts_api.h, 983
VTSS_PHY_TS_EGR_RW_FCS_ERR
vtss_phy_ts_api.h, 981
VTSS_PHY_TS_EGR_TIMESTAMP_CAPTURED
vtss_phy_ts_api.h, 981
VTSS_PHY_TS_ENG_FLOW_VALID_FOR_CH0
vtss_phy_ts_api.h, 979
VTSS_PHY_TS_ENG_FLOW_VALID_FOR_CH1
vtss_phy_ts_api.h, 980
VTSS_PHY_TS_ETH_ADDR_MATCH_48BIT
vtss_phy_ts_api.h, 975
VTSS_PHY_TS_ETH_ADDR_MATCH_ANY_MULTICAST
vtss_phy_ts_api.h, 975
VTSS_PHY_TS_ETH_ADDR_MATCH_ANY_UNICAST
vtss_phy_ts_api.h, 975
VTSS_PHY_TS_ETH_MATCH_DEST_ADDR
vtss_phy_ts_api.h, 975
VTSS_PHY_TS_ETH_MATCH_SRC_ADDR
vtss_phy_ts_api.h, 975
VTSS_PHY_TS_ETH_MATCH_SRC_OR_DEST
vtss_phy_ts_api.h, 976
VTSS_PHY_TS_FIFO_SIG_DEST_IP
vtss_phy_ts_api.h, 971
VTSS_PHY_TS_FIFO_SIG_DEST_MAC
vtss_phy_ts_api.h, 972
VTSS_PHY_TS_FIFO_SIG_DOMAIN_NUM
vtss_phy_ts_api.h, 972
VTSS_PHY_TS_FIFO_SIG_MSG_TYPE
vtss_phy_ts_api.h, 971
VTSS_PHY_TS_FIFO_SIG_SEQ_ID
vtss_phy_ts_api.h, 972
VTSS_PHY_TS_FIFO_SIG_SOURCE_PORT_ID
vtss_phy_ts_api.h, 972
VTSS_PHY_TS_FIFO_SIG_SRC_IP
vtss_phy_ts_api.h, 971
VTSS_PHY_TS_INGR_DATAPATH_RESET
vtss_phy_ts_api.h, 982
VTSS_PHY_TS_INGR_ENGINE_ERR
vtss_phy_ts_api.h, 980

VTSS_PHY_TS_INGR_LTC1_RESET
 vtss_phy_ts_api.h, 983

VTSS_PHY_TS_INGR_RW_FCS_ERR
 vtss_phy_ts_api.h, 980

VTSS_PHY_TS_INGR_RW_PREAM_ERR
 vtss_phy_ts_api.h, 980

VTSS_PHY_TS_IP_MATCH_DEST
 vtss_phy_ts_api.h, 978

VTSS_PHY_TS_IP_MATCH_SRC_OR_DEST
 vtss_phy_ts_api.h, 978

VTSS_PHY_TS_IP_MATCH_SRC
 vtss_phy_ts_api.h, 978

VTSS_PHY_TS_IP_VER_4
 vtss_phy_ts_api.h, 977

VTSS_PHY_TS_IP_VER_6
 vtss_phy_ts_api.h, 977

VTSS_PHY_TS_LTC_LOAD_SAVE_NEW_TOD
 vtss_phy_ts_api.h, 982

VTSS_PHY_TS_LTC_NEW_PPS_INTRPT
 vtss_phy_ts_api.h, 982

VTSS_PHY_TS_MPLS_STACK_DEPTH_1
 vtss_phy_ts_api.h, 978

VTSS_PHY_TS_MPLS_STACK_DEPTH_2
 vtss_phy_ts_api.h, 978

VTSS_PHY_TS_MPLS_STACK_DEPTH_3
 vtss_phy_ts_api.h, 979

VTSS_PHY_TS_MPLS_STACK_DEPTH_4
 vtss_phy_ts_api.h, 979

VTSS_PHY_TS_MPLS_STACK_REF_POINT_END
 vtss_phy_ts_api.h, 979

VTSS_PHY_TS_MPLS_STACK_REF_POINT_TOP
 vtss_phy_ts_api.h, 979

VTSS_PHY_TS_SIG_DEST_IP_LEN
 vtss_phy_ts_api.h, 974

VTSS_PHY_TS_SIG_DEST_MAC_LEN
 vtss_phy_ts_api.h, 974

VTSS_PHY_TS_SIG_DOMAIN_NUM_LEN
 vtss_phy_ts_api.h, 973

VTSS_PHY_TS_SIG_LEN
 vtss_phy_ts_api.h, 972

VTSS_PHY_TS_SIG_MSG_TYPE_LEN
 vtss_phy_ts_api.h, 973

VTSS_PHY_TS_SIG_SEQ_ID_LEN
 vtss_phy_ts_api.h, 973

VTSS_PHY_TS_SIG_SEQUENCE_ID_LEN
 vtss_phy_ts_api.h, 974

VTSS_PHY_TS_SIG_SOURCE_PORT_ID_LEN
 vtss_phy_ts_api.h, 973

VTSS_PHY_TS_SIG_SRC_IP_LEN
 vtss_phy_ts_api.h, 974

VTSS_PHY_TS_SIG_TIME_STAMP_LEN
 vtss_phy_ts_api.h, 973

VTSS_PHY_TS_SPI_CLK_THRU_PPS0
 options.h, 541

VTSS_PHY_TS_TAG_RANGE_INNER
 vtss_phy_ts_api.h, 977

VTSS_PHY_TS_TAG_RANGE_NONE
 vtss_phy_ts_api.h, 977

VTSS_PHY_TS_TAG_RANGE_OUTER
 vtss_phy_ts_api.h, 977

VTSS_PHY_TS_TAG_TYPE_B
 vtss_phy_ts_api.h, 976

VTSS_PHY_TS_TAG_TYPE_C
 vtss_phy_ts_api.h, 976

VTSS_PHY_TS_TAG_TYPE_I
 vtss_phy_ts_api.h, 976

VTSS_PHY_TS_TAG_TYPE_S
 vtss_phy_ts_api.h, 976

VTSS_PORT_ARRAY_SIZE
 types.h, 582

VTSS_PORT_COUNT
 types.h, 581

VTSS_PORT_IS_PORT
 types.h, 582

VTSS_PORT_NO_ANY
 vtss_security_api.h, 1060

VTSS_PORT_NO_CPU
 types.h, 582

VTSS_PORT_NO_END
 types.h, 582

VTSS_PORT_NO_NONE
 types.h, 581

VTSS_PORT_NO_START
 types.h, 582

VTSS_PORT_POLICER_CPU_QUEUES
 vtss_qos_api.h, 1050

VTSS_PORT_POLICERS
 vtss_qos_api.h, 1050

VTSS_PORTS
 types.h, 581

VTSS_PRIO_ARRAY_SIZE
 types.h, 583

VTSS_PRIO_END
 types.h, 583

VTSS_PRIO_NO_NONE
 types.h, 583

VTSS_PRIO_START
 types.h, 583

VTSS_PRIO_SUPER
 vtss_packet_api.h, 773

VTSS_PRIOS
 types.h, 583

VTSS_PTP_IP_1588_VERSION_2_1
 vtss_phy_ts_api.h, 974

VTSS_PVLAN_ARRAY_SIZE
 vtss_l2_api.h, 652

VTSS_PVLAN_NO_DEFAULT
 vtss_l2_api.h, 652

VTSS_PVLAN_NO_END
 vtss_l2_api.h, 652

VTSS_PVLAN_NO_START
 vtss_l2_api.h, 652

VTSS_PVLANS
 vtss_l2_api.h, 651

VTSS_QCE_ID_LAST
 vtss_qos_api.h, 1051

VTSS_QCL_ARRAY_SIZE
 vtss_qos_api.h, 1051

VTSS_QCL_ID_END
 vtss_qos_api.h, 1051

VTSS_QCL_ID_START
 vtss_qos_api.h, 1051

VTSS_QCL_IDS
 vtss_qos_api.h, 1050

VTSS_QS_CONF_MAX
 vtss_init_api.h, 625

VTSS_QS_CONF_MIN
 vtss_init_api.h, 625

VTSS_QUEUE_ARRAY_SIZE
 types.h, 584

VTSS_QUEUE_END
 types.h, 584

VTSS_QUEUE_START
 types.h, 584

VTSS_QUEUES
 types.h, 584

VTSS_RLEG_CNT
 vtss_l3_api.h, 705

VTSS_SEC_NS_INTERVAL
 types.h, 597

VTSS_SLEWRATE_120PS
 vtss_phy_10g_api.h, 807

VTSS_SLEWRATE_25PS
 vtss_phy_10g_api.h, 806

VTSS_SLEWRATE_35PS
 vtss_phy_10g_api.h, 807

VTSS_SLEWRATE_55PS
 vtss_phy_10g_api.h, 807

VTSS_SLEWRATE_70PS
 vtss_phy_10g_api.h, 807

VTSS_SLEWRATE_INVALID
 vtss_phy_10g_api.h, 807

VTSS_SVL_PACKET_HDR_SIZE_BYTES
 vtss_packet_api.h, 774

VTSS_SVL_RX_IFH_SIZE
 vtss_packet_api.h, 774

VTSS_SYNCE_CLK_MAX
 vtss_sync_api.h, 1068

VTSS_SYNCE_CLK_PORT_ARRAY_SIZE
 types.h, 597

VTSS_SYNCE_CLK_A
 vtss_sync_api.h, 1068

VTSS_SYNCE_CLK_B
 vtss_sync_api.h, 1068

VTSS_TIME_OF_DAY
 vtss_os_ecos.h, 754

 vtss_os_linux.h, 765

VTSS_UPSPN_CPU
 vtss_l2_api.h, 647

VTSS_UPSPN_NONE
 vtss_l2_api.h, 647

VTSS_VCE_ID_LAST
 vtss_l2_api.h, 649

VTSS_VCL_ARRAY_SIZE
 vtss_l2_api.h, 649

VTSS_VCL_ID_END
 vtss_l2_api.h, 648

VTSS_VCL_ID_START
 vtss_l2_api.h, 648

VTSS_VCL_IDS
 vtss_l2_api.h, 648

VTSS_VID_ALL
 types.h, 588

VTSS_VID_DEFAULT
 types.h, 587

VTSS_VID_NULL
 types.h, 587

VTSS_VID_RESERVED
 types.h, 588

VTSS_VIDS
 types.h, 588

VTSS_VLAN_TRANS_FIRST_GROUP_ID
 vtss_l2_api.h, 650

VTSS_VLAN_TRANS_GROUP_MAX_CNT
 vtss_l2_api.h, 649

VTSS_VLAN_TRANS_LAST_GROUP_ID
 vtss_l2_api.h, 650

VTSS_VLAN_TRANS_MAX_CNT
 vtss_l2_api.h, 649

VTSS_VLAN_TRANS_MAX_VLAN_ID
 vtss_l2_api.h, 650

VTSS_VLAN_TRANS_NULL_CHECK
 vtss_l2_api.h, 651

VTSS_VLAN_TRANS_NULL_GROUP_ID
 vtss_l2_api.h, 649

VTSS_VLAN_TRANS_PORT_BF_SIZE
 vtss_l2_api.h, 651

VTSS_VLAN_TRANS_VALID_GROUP_CHECK
 vtss_l2_api.h, 650

VTSS_VLAN_TRANS_VALID_VLAN_CHECK
 vtss_l2_api.h, 651

VTSS_VLAN_TRANS_VID_START
 vtss_l2_api.h, 650

VTSS_VSTAX_HDR_SIZE
 vtss_packet_api.h, 773

VTSS_VSTAX_TTL_PORT
 vtss_packet_api.h, 773

VTSS_VSTAX_UPSID_LEGAL
 vtss_l2_api.h, 646

VTSS_VSTAX_UPSID_MAX
 vtss_l2_api.h, 646

VTSS_VSTAX_UPSID_MIN
 vtss_l2_api.h, 646

VTSS_VSTAX_UPSID_START
 vtss_l2_api.h, 646

VTSS_VSTAX_UPSID_UNDEF
 vtss_l2_api.h, 647

VTSS_VSTAX_UPSIDS
 vtss_l2_api.h, 646

val
 vtss_eee_port_state_t, 89

 vtss_phy_conf_1g_t, 315

vtss_phy_ts_eth_conf_t, 356, 357
 vtss_phy_ts_ptp_conf_t, 386
 vtss_phy_ts_y1731_oam_conf_t, 395
valid
 vtss_ewis_tti_s, 129
 vtss_vstax_rx_header_t, 532
value
 ib_par_cfg, 31
 vtss_phy_ts_ach_conf_t, 344
 vtss_phy_ts_eth_conf_t, 356, 357
 vtss_phy_ts_ptp_conf_t, 387
 vtss_phy_ts_y1731_oam_conf_t, 396
 vtss_sgpi_port_data_t, 477
 vtss_vcap_ip_t, 492
 vtss_vcap_u128_t, 493
 vtss_vcap_u16_t, 494
 vtss_vcap_u24_t, 495
 vtss_vcap_u32_t, 496
 vtss_vcap_u40_t, 497
 vtss_vcap_u48_t, 498
 vtss_vcap_u8_t, 499
 vtss_vcap_vid_t, 501
 vtss_vcap_vr_t, 503
venice_rev_a_los_detection_workaround
 vtss_phy_10g_mode_t, 270
version
 vtss_phy_ts_ach_conf_t, 344
 vtss_phy_ts_oam_engine_action_t, 382
vga_adc_debug
 vtss_phy_api.h, 935
vid
 vtss_ace_vlan_t, 56
 vtss_ece_inner_tag_t, 77
 vtss_ece_outer_tag_t, 82
 vtss_ece_tag_t, 84
 vtss_evc_pb_conf_t, 95
 vtss_mirror_conf_t, 171
 vtss_packet_frame_info_t, 176
 vtss_packet_port_info_t, 179
 vtss_qce_tag_t, 454
 vtss_tci_t, 482
 vtss_vce_action_t, 504
 vtss_vce_tag_t, 516
 vtss_vid_mac_t, 518
 vtss_vlan_tag_t, 522
 vtss_vlan_trans_grp2vlan_conf_t, 523
vid_mac
 vtss_mac_table_entry_t, 166
vlan
 vtss_ace_t, 53
 vtss_l3_neighbour_t, 159
 vtss_l3_rleg_conf_t, 161
 vtss_routing_entry_t, 471
vlan_check
 vtss_phy_ts_eth_conf_t, 354
vml_format
 vtss_debug_info_t, 69
vp
 vtss_phy_10g_ob_status_t, 275
vr
 vtss_vcap_vr_t, 504
vrid0
 vtss_l3_rleg_conf_t, 161
vrid0_enable
 vtss_l3_rleg_conf_t, 161
vrid1
 vtss_l3_rleg_conf_t, 162
vrid1_enable
 vtss_l3_rleg_conf_t, 162
vstax
 vtss_packet_rx_header_t, 183
 vtss_packet_rx_info_t, 192
 vtss_packet_tx_info_t, 211
vstax2
 vtss_mac_table_entry_t, 168
vtaill
 vtss_phy_10g_jitter_conf_t, 252
vtss_10g_phy_gpio_t
 vtss_phy_10g_api.h, 837
vtss_32_cntr_t
 vtss_phy_10g_api.h, 821
vtss_ace_add
 vtss_security_api.h, 1065
vtss_ace_bit_t
 vtss_security_api.h, 1061
vtss_ace_counter_clear
 vtss_security_api.h, 1066
vtss_ace_counter_get
 vtss_security_api.h, 1066
vtss_ace_del
 vtss_security_api.h, 1065
vtss_ace_frame_arp_t, 36
 arp, 37
 dip, 39
 dmac_match, 38
 ethernet, 38
 ip, 38
 length, 38
 req, 37
 sip, 38
 smac, 37
 smac_match, 37
 unknown, 37
vtss_ace_frame_etype_t, 39
 data, 40
 dmac, 39
 etype, 40
 smac, 40
vtss_ace_frame_ipv4_t, 40
 data, 42
 dip, 42
 dport, 43
 ds, 42
 fragment, 41
 options, 41
 proto, 42

seq_zero, 45
sip, 42
sip_eq_dip, 44
sport, 43
sport_eq_dport, 44
tcp_ack, 44
tcp_fin, 43
tcp_psh, 44
tcp_rst, 43
tcp_syn, 43
tcp_urg, 44
ttl, 41
vtss_ace_frame_ipv6_t, 45
 data, 46
 dport, 47
 ds, 46
 proto, 46
 seq_zero, 49
 sip, 46
 sip_eq_dip, 48
 sport, 47
 sport_eq_dport, 48
 tcp_ack, 48
 tcp_fin, 47
 tcp_psh, 48
 tcp_rst, 47
 tcp_syn, 47
 tcp_urg, 48
 ttl, 46
vtss_ace_frame_llc_t, 49
 dmac, 49
 llc, 50
 smac, 50
vtss_ace_frame_snap_t, 50
 dmac, 51
 smac, 51
 snap, 51
vtss_ace_init
 vtss_security_api.h, 1065
vtss_ace_t, 51
 action, 53
 arp, 54
 dmac_bc, 53
 dmac_mc, 53
 etype, 54
 frame, 55
 id, 52
 ipv4, 54
 ipv6, 55
 llc, 54
 policy, 52
 port_no, 52
 snap, 54
 type, 53
 vlan, 53
vtss_ace_type_t
 vtss_security_api.h, 1060
vtss_ace_vlan_t, 55
 cfi, 56
 usr_prio, 56
 vid, 56
 vtss_acl_action_t, 56
 cpu, 57
 cpu_once, 57
 cpu_queue, 57
 forward, 58
 irq_trigger, 59
 learn, 58
 police, 57
 policer_no, 58
 port_forward, 58
 port_no, 58
 vtss_acl_policer_conf_get
 vtss_security_api.h, 1062
 vtss_acl_policer_conf_set
 vtss_security_api.h, 1062
 vtss_acl_policer_conf_t, 59
 rate, 59
 vtss_acl_port_conf_get
 vtss_security_api.h, 1063
 vtss_acl_port_conf_set
 vtss_security_api.h, 1063
 vtss_acl_port_conf_t, 60
 action, 60
 policy_no, 60
 vtss_acl_port_counter_clear
 vtss_security_api.h, 1064
 vtss_acl_port_counter_get
 vtss_security_api.h, 1064
 vtss_aggr_glag_members_get
 vtss_l2_api.h, 685
 vtss_aggr_mode_get
 vtss_l2_api.h, 685
 vtss_aggr_mode_set
 vtss_l2_api.h, 685
 vtss_aggr_mode_t, 61
 dmac_enable, 61
 sip_dip_enable, 62
 smac_enable, 61
 sport_dport_enable, 62
 vtss_aggr_port_members_get
 vtss_l2_api.h, 684
 vtss_aggr_port_members_set
 vtss_l2_api.h, 684
 vtss_aneg_t, 62
 generate_pause, 63
 obey_pause, 63
 vtss_api/include/vtss/api/l2_types.h, 537
 vtss_api/include/vtss/api/options.h, 538
 vtss_api/include/vtss/api/phy.h, 556
 vtss_api/include/vtss/api/port.h, 558
 vtss_api/include/vtss/api/types.h, 571
 vtss_api/include/vtss_ae_api.h, 609
 vtss_api/include/vtss_afi_api.h, 609
 vtss_api/include/vtss_aneg_api.h, 610
 vtss_api/include/vtss_api.h, 610

vtss_api/include/vtss_evc_api.h, 610
 vtss_api/include/vtss_fdma_api.h, 621
 vtss_api/include/vtss_gfp_api.h, 621
 vtss_api/include/vtss_hqos_api.h, 621
 vtss_api/include/vtss_i2c_api.h, 622
 vtss_api/include/vtss_init_api.h, 622
 vtss_api/include/vtss_l2_api.h, 637
 vtss_api/include/vtss_l3_api.h, 702
 vtss_api/include/vtss_mac10g_api.h, 714
 vtss_api/include/vtss_misc_api.h, 714
 vtss_api/include/vtss_mpls_api.h, 746
 vtss_api/include/vtss_oam_api.h, 746
 vtss_api/include/vtss_oha_api.h, 747
 vtss_api/include/vtss_os.h, 747
 vtss_api/include/vtss_os_custom.h, 747
 vtss_api/include/vtss_os_ecos.h, 752
 vtss_api/include/vtss_os_linux.h, 762
 vtss_api/include/vtss_otn_api.h, 769
 vtss_api/include/vtss_packet_api.h, 769
 vtss_api/include/vtss_pcs_10gbase_r_api.h, 792
 vtss_api/include/vtss_phy_10g_api.h, 792
 vtss_api/include/vtss_phy_api.h, 886
 vtss_api/include/vtss_phy_ts_api.h, 962
 vtss_api/include/vtss_port_api.h, 1031
 vtss_api/include/vtss_qos_api.h, 1048
 vtss_api/include/vtss_rab_api.h, 1057
 vtss_api/include/vtss_security_api.h, 1057
 vtss_api/include/vtss_sf14_api.h, 1067
 vtss_api/include/vtss_sync_api.h, 1067
 vtss_api/include/vtss_tfi5_api.h, 1071
 vtss_api/include/vtss_ts_api.h, 1071
 vtss_api/include/vtss_upi_api.h, 1088
 vtss_api/include/vtss_wis_api.h, 1088
 vtss_api/include/vtss_xaui_api.h, 1118
 vtss_api/include/vtss_xfi_api.h, 1118
 vtss_api_lock_t, 63
 file, 64
 function, 64
 inst, 64
 line, 64
 vtss_apvlan_port_members_get
 vtss_l2_api.h, 681
 vtss_apvlan_port_members_set
 vtss_l2_api.h, 681
 vtss_auth_port_state_get
 vtss_security_api.h, 1061
 vtss_auth_port_state_set
 vtss_security_api.h, 1062
 vtss_auth_state_t
 vtss_security_api.h, 1060
 vtss_basic_counters_t, 65
 rx_frames, 65
 tx_frames, 65
 vtss_callout_free
 vtss_os_ecos.h, 761
 vtss_callout_lock
 vtss_misc_api.h, 728
 vtss_callout_malloc
 vtss_os_ecos.h, 761
 vtss_callout_trace_hex_dump
 vtss_misc_api.h, 727
 vtss_callout_trace_printf
 vtss_misc_api.h, 726
 vtss_callout_unlock
 vtss_misc_api.h, 728
 vtss_channel_t
 vtss_phy_10g_api.h, 828
 vtss_chip_id_get
 vtss_misc_api.h, 731
 vtss_chip_id_t, 65
 part_number, 66
 revision, 66
 vtss_ckout_data_sel_t
 vtss_phy_10g_api.h, 829
 vtss_counter_pair_t, 66
 bytes, 67
 frames, 67
 vtss_debug_group_t
 vtss_misc_api.h, 722
 vtss_debug_info_get
 vtss_misc_api.h, 727
 vtss_debug_info_print
 vtss_misc_api.h, 728
 vtss_debug_info_t, 67
 chip_no, 68
 clear, 69
 full, 68
 group, 68
 layer, 68
 port_list, 68
 vml_format, 69
 vtss_debug_layer_t
 vtss_misc_api.h, 722
 vtss_debug_lock
 vtss_misc_api.h, 729
 vtss_debug_lock_t, 69
 chip_no, 70
 vtss_debug_reg_check_set
 vtss_misc_api.h, 745
 vtss_debug_unlock
 vtss_misc_api.h, 729
 vtss_dev_all_event_enable
 vtss_misc_api.h, 733
 vtss_dev_all_event_poll
 vtss_misc_api.h, 733
 vtss_dgroup_port_conf_get
 vtss_l2_api.h, 681
 vtss_dgroup_port_conf_set
 vtss_l2_api.h, 682
 vtss_dgroup_port_conf_t, 70
 dgroup_no, 70
 vtss_dlb_policer_conf_t, 71
 cbs, 72
 cf, 72
 cir, 72
 cm, 71

ebs, 73
eir, 72
enable, 71
line_rate, 72
type, 71
vtss_dscp_emode_t
 vtss_qos_api.h, 1052
vtss_dscp_mode_t
 vtss_qos_api.h, 1052
vtss_ece_action_t, 73
 dir, 73
 evc_id, 74
 inner_tag, 74
 outer_tag, 74
 policer_id, 74
 policy_no, 75
 pop_tag, 74
vtss_ece_add
 vtss_evc_api.h, 618
vtss_ece_counters_clear
 vtss_evc_api.h, 620
vtss_ece_counters_get
 vtss_evc_api.h, 620
vtss_ece_del
 vtss_evc_api.h, 618
vtss_ece_dir_t
 types.h, 607
vtss_ece_frame_ipv4_t, 75
 dscp, 75
vtss_ece_frame_ipv6_t, 76
 dscp, 76
vtss_ece_init
 vtss_evc_api.h, 617
vtss_ece_inner_tag_t, 77
 dei, 78
 pcp, 78
 pcp_dei_preserve, 77
 type, 77
 vid, 77
vtss_ece_inner_tag_type_t
 types.h, 608
vtss_ece_key_t, 78
 frame, 80
 inner_tag, 79
 ipv4, 80
 ipv6, 80
 mac, 79
 port_list, 79
 tag, 79
 type, 79
vtss_ece_mac_t, 80
 dmac, 81
vtss_ece_outer_tag_t, 81
 dei, 82
 enable, 82
 pcp, 82
 pcp_dei_preserve, 82
 vid, 82
vtss_ece_pop_tag_t
 types.h, 608
vtss_ece_port_t
 vtss_evc_api.h, 614
vtss_ece_t, 83
 action, 84
 id, 83
 key, 83
vtss_ece_tag_t, 84
 dei, 85
 pcp, 85
 s_tagged, 85
 tagged, 85
 vid, 84
vtss_ece_type_t
 vtss_evc_api.h, 614
vtss_eee_port_conf_set
 vtss_misc_api.h, 744
vtss_eee_port_conf_t, 86
 eee_ena, 86
 eee_fast_queues, 86
 lp_advertisement, 87
 optimized_for_power, 87
 tx_tw, 86
vtss_eee_port_counter_get
 vtss_misc_api.h, 745
vtss_eee_port_counter_t, 87
 fill_level, 88
 fill_level_get, 88
 fill_level_thres, 88
 tx_out_bytes, 88
 tx_out_bytes_get, 88
vtss_eee_port_state_set
 vtss_misc_api.h, 744
vtss_eee_port_state_t, 89
 select, 89
 val, 89
vtss_eee_state_select_t
 vtss_misc_api.h, 725
vtss_eps_port_conf_get
 vtss_l2_api.h, 696
vtss_eps_port_conf_set
 vtss_l2_api.h, 696
vtss_eps_port_conf_t, 90
 port_no, 90
 type, 90
vtss_eps_port_selector_get
 vtss_l2_api.h, 697
vtss_eps_port_selector_set
 vtss_l2_api.h, 697
vtss_eps_port_type_t
 vtss_l2_api.h, 655
vtss_eps_selector_t
 vtss_l2_api.h, 655
vtss_erps_port_state_get
 vtss_l2_api.h, 699
vtss_erps_port_state_set
 vtss_l2_api.h, 699

vtss_erps_state_t
 vtss_l2_api.h, 656
 vtss_erps_vlan_member_get
 vtss_l2_api.h, 698
 vtss_erps_vlan_member_set
 vtss_l2_api.h, 698
 vtss_evc_add
 vtss_evc_api.h, 616
 vtss_evc_api.h
 VTSS_ECE_ID_LAST, 613
 VTSS_EVC_ID_NONE, 613
 VTSS_EVC_POLICER_ID_DISCARD, 613
 VTSS_EVC_POLICER_ID_EVC, 613
 VTSS_EVC_POLICER_ID_NONE, 613
 VTSS_EVC_POLICERS, 612
 vtss_ece_add, 618
 vtss_ece_counters_clear, 620
 vtss_ece_counters_get, 620
 vtss_ece_del, 618
 vtss_ece_init, 617
 vtss_ece_port_t, 614
 vtss_ece_type_t, 614
 vtss_evc_add, 616
 vtss_evc_counters_clear, 619
 vtss_evc_counters_get, 619
 vtss_evc_del, 617
 vtss_evc_get, 617
 vtss_evc_policer_conf_get, 615
 vtss_evc_policer_conf_set, 616
 vtss_evc_port_conf_get, 614
 vtss_evc_port_conf_set, 615
 vtss_evc_conf_t, 91
 learning, 91
 network, 92
 pb, 92
 policer_id, 91
 vtss_evc_counters_clear
 vtss_evc_api.h, 619
 vtss_evc_counters_get
 vtss_evc_api.h, 619
 vtss_evc_counters_t, 92
 rx_discard, 93
 rx_green, 93
 rx_red, 93
 rx_yellow, 93
 tx_discard, 93
 tx_green, 94
 tx_yellow, 94
 vtss_evc_del
 vtss_evc_api.h, 617
 vtss_evc_get
 vtss_evc_api.h, 617
 vtss_evc_pb_conf_t, 94
 ivid, 95
 leaf, 96
 leaf_ivid, 95
 leaf_vid, 95
 nni, 95
 vid, 95
 vtss_evc_policer_conf_get
 vtss_evc_api.h, 615
 vtss_evc_policer_conf_set
 vtss_evc_api.h, 616
 vtss_evc_port_conf_get
 vtss_evc_api.h, 614
 vtss_evc_port_conf_set
 vtss_evc_api.h, 615
 vtss_evc_port_conf_t, 96
 dei_colouring, 96
 vtss_ewis_aisl_cons_act_s, 97
 ais_on_lof, 97
 ais_on_los, 97
 vtss_ewis_conf_s, 98
 ewis_cntr_thresh_conf, 101
 ewis_init_done, 98
 ewis_mode, 99
 exp_sl, 100
 force_mode, 99
 path_txti, 100
 perf_mode, 101
 section_cons_act, 99
 section_txti, 99
 static_conf, 99
 test_conf, 100
 tx_oh, 100
 tx_oh_passthru, 100
 vtss_ewis_cons_act_get
 vtss_wis_api.h, 1109
 vtss_ewis_cons_act_s, 101
 aisl, 102
 fault, 102
 rdil, 102
 vtss_ewis_cons_act_set
 vtss_wis_api.h, 1108
 vtss_ewis_counter_get
 vtss_wis_api.h, 1115
 vtss_ewis_counter_s, 102
 pf_ebc_l, 103
 pf_ebc_p, 103
 pn_ebc_l, 103
 pn_ebc_p, 103
 pn_ebc_s, 103
 vtss_ewis_counter_threshold_get
 vtss_wis_api.h, 1117
 vtss_ewis_counter_threshold_s, 104
 f_ebc_thr_l, 105
 f_ebc_thr_p, 105
 n_ebc_thr_l, 104
 n_ebc_thr_p, 105
 n_ebc_thr_s, 104
 vtss_ewis_counter_threshold_set
 vtss_wis_api.h, 1116
 vtss_ewis_defects_get
 vtss_wis_api.h, 1113
 vtss_ewis_defects_s, 105
 dais_l, 107

dais_p, 107
dfais_p, 108
dfplm_p, 108
dfuneq_p, 109
dlcd_p, 108
dl/of_s, 106
dl/op_p, 107
dl/os_s, 106
doof_s, 106
dplm_p, 108
drdi_l, 107
drdi_p, 108
duneq_p, 107
vtss_ewis_event_enable
 vtss_wis_api.h, 1102
vtss_ewis_event_force
 vtss_wis_api.h, 1103
vtss_ewis_event_poll
 vtss_wis_api.h, 1102
vtss_ewis_event_poll_without_mask
 vtss_wis_api.h, 1103
vtss_ewis_event_t
 vtss_wis_api.h, 1100
vtss_ewis_exp_sl_set
 vtss_wis_api.h, 1110
vtss_ewis_fault_cons_act_s, 109
 fault_on_aisl, 111
 fault_on_aisp, 111
 fault_on_feaisp, 110
 fault_on_feplmp, 110
 fault_on_lcdp, 111
 fault_on_lof, 110
 fault_on_lopp, 111
 fault_on_los, 110
 fault_on_plmp, 111
 fault_on_rdil, 110
 fault_on_sef, 110
vtss_ewis_force_conf_get
 vtss_wis_api.h, 1105
vtss_ewis_force_conf_set
 vtss_wis_api.h, 1104
vtss_ewis_force_mode_s, 112
 line_rx_force, 112
 line_tx_force, 112
 path_force, 113
vtss_ewis_line_force_mode_s, 113
 force_ais, 113
 force_rdi, 114
vtss_ewis_line_tx_force_mode_s, 114
 force_ais, 114
 force_rdi, 115
vtss_ewis_mode_get
 vtss_wis_api.h, 1108
vtss_ewis_mode_set
 vtss_wis_api.h, 1107
vtss_ewis_mode_t
 vtss_wis_api.h, 1101
vtss_ewis_path_acti_get
 vtss_wis_api.h, 1115
vtss_ewis_path_force_mode_s, 115
 force_rdi, 116
 force_uneq, 115
vtss_ewis_path_txti_get
 vtss_wis_api.h, 1111
vtss_ewis_path_txti_set
 vtss_wis_api.h, 1111
vtss_ewis_perf_cntr_mode_t
 vtss_wis_api.h, 1101
vtss_ewis_perf_get
 vtss_wis_api.h, 1116
vtss_ewis_perf_mode_get
 vtss_wis_api.h, 1117
vtss_ewis_perf_mode_s, 116
 pf_ebc_mode_l, 117
 pf_ebc_mode_p, 117
 pn_ebc_mode_l, 117
 pn_ebc_mode_p, 117
 pn_ebc_mode_s, 116
vtss_ewis_perf_mode_set
 vtss_wis_api.h, 1117
vtss_ewis_perf_s, 118
 pf_ebc_l, 118
 pf_ebc_p, 119
 pn_ebc_l, 118
 pn_ebc_p, 119
 pn_ebc_s, 118
vtss_ewis_prbs31_err_inj_set
 vtss_wis_api.h, 1112
vtss_ewis_prbs31_err_inj_t
 vtss_wis_api.h, 1102
vtss_ewis_rdil_cons_act_s, 119
 rdil_on_ais_l, 120
 rdil_on_lof, 120
 rdil_on_lopc, 120
 rdil_on_los, 120
vtss_ewis_reset
 vtss_wis_api.h, 1108
vtss_ewis_section_acti_get
 vtss_wis_api.h, 1114
vtss_ewis_section_txti_get
 vtss_wis_api.h, 1110
vtss_ewis_section_txti_set
 vtss_wis_api.h, 1109
vtss_ewis_sl_conf_s, 121
 exsl, 121
vtss_ewis_static_conf_get
 vtss_wis_api.h, 1104
vtss_ewis_static_conf_s, 121
 ewis_cnt_cfg, 125
 ewis_lof_ctrl1, 124
 ewis_lof_ctrl2, 124
 ewis_mode_ctrl, 123
 ewis_pmtick_ctrl, 125
 ewis_rx_ctrl1, 122
 ewis_rx_err_frc1, 124
 ewis_rx_frm_ctrl1, 124

ewis_rx_frm_ctrl2, 124
 ewis_tx_a1_a2, 123
 ewis_tx_c2_h1, 123
 ewis_tx_h2_h3, 123
 ewis_tx_z0_e1, 123
 ewis_txctrl1, 122
 ewis_txctrl2, 122
 vtss_ewis_static_conf_t
 vtss_wis_api.h, 1100
 vtss_ewis_status_get
 vtss_wis_api.h, 1114
 vtss_ewis_status_s, 125
 fault, 126
 link_stat, 126
 vtss_ewis_test_conf_s, 126
 loopback, 127
 test_pattern_ana, 127
 test_pattern_gen, 127
 vtss_ewis_test_counter_get
 vtss_wis_api.h, 1113
 vtss_ewis_test_mode_get
 vtss_wis_api.h, 1112
 vtss_ewis_test_mode_set
 vtss_wis_api.h, 1111
 vtss_ewis_test_pattern_s
 vtss_wis_api.h, 1101
 vtss_ewis_test_status_s, 127
 ana_sync, 128
 tstpat_cnt, 128
 vtss_ewis_tti_mode_t
 vtss_wis_api.h, 1100
 vtss_ewis_tti_s, 128
 mode, 129
 tti, 129
 valid, 129
 vtss_ewis_tx_oh_get
 vtss_wis_api.h, 1106
 vtss_ewis_tx_oh_passthru_get
 vtss_wis_api.h, 1107
 vtss_ewis_tx_oh_passthru_set
 vtss_wis_api.h, 1106
 vtss_ewis_tx_oh_s, 129
 tx_c2, 132
 tx_dcc_l, 131
 tx_dcc_s, 130
 tx_e1, 130
 tx_e2, 131
 tx_f1, 130
 tx_f2, 132
 tx_k1_k2, 131
 tx_n1, 132
 tx_s1, 131
 tx_z0, 131
 tx_z1_z2, 132
 tx_z3_z4, 132
 vtss_ewis_tx_oh_set
 vtss_wis_api.h, 1105
 vtss_ewis_tx_passthru_s, 133
 tx_b1, 134
 tx_b2, 135
 tx_dcc_l, 136
 tx_dcc_s, 134
 tx_e1, 134
 tx_e2, 136
 tx_f1, 134
 tx_j0, 133
 tx_k1, 135
 tx_k2, 135
 tx_loh, 136
 tx_reil, 135
 tx_s1, 136
 tx_soh, 135
 tx_z0, 134
 tx_z1_z2, 136
 vtss_fan_conf_t, 137
 fan_low_pol, 137
 fan_open_col, 138
 fan_pwm_freq, 137
 ppr, 138
 type, 138
 vtss_fan_controller_init
 vtss_misc_api.h, 743
 vtss_fan_cool_lvl_get
 vtss_misc_api.h, 744
 vtss_fan_cool_lvl_set
 vtss_misc_api.h, 743
 vtss_fan_rotation_get
 vtss_misc_api.h, 742
 vtss_fefi_mode_t
 vtss_phy_api.h, 915
 vtss_fiber_port_speed_t
 port.h, 571
 vtss_gpio_10g_aggr_intrpt_t
 vtss_phy_10g_api.h, 841
 vtss_gpio_10g_chan_intrpt_t
 vtss_phy_10g_api.h, 840
 vtss_gpio_10g_gpio_intr_sgnl_t
 vtss_phy_10g_api.h, 838
 vtss_gpio_10g_gpio_mode_t, 138
 aggr_intrpt, 140
 c_intrpt, 140
 in_sig, 140
 input, 139
 mode, 139
 p_gpio, 140
 p_gpio_intrpt, 141
 port, 139
 source, 140
 use_as_intrpt, 141
 vtss_gpio_10g_input_t
 vtss_phy_10g_api.h, 841
 vtss_gpio_10g_no_t
 vtss_phy_10g_api.h, 821
 vtss_gpio_direction_set
 vtss_misc_api.h, 734
 vtss_gpio_event_enable

vtss_misc_api.h, 736
vtss_gpio_event_poll
 vtss_misc_api.h, 736
vtss_gpio_mode_set
 vtss_misc_api.h, 734
vtss_gpio_mode_t
 vtss_misc_api.h, 723
vtss_gpio_read
 vtss_misc_api.h, 735
vtss_gpio_write
 vtss_misc_api.h, 735
vtss_hqos_sch_mode_t
 types.h, 609
vtss_i2c_read_t
 vtss_init_api.h, 626
vtss_i2c_write_t
 vtss_init_api.h, 627
vtss_init_api.h
 VTSS_I2C_NO_MULTIPLEXER, 625
 VTSS_QS_CONF_MAX, 625
 VTSS_QS_CONF_MIN, 625
 vtss_i2c_read_t, 626
 vtss_i2c_write_t, 627
 vtss_init_conf_get, 634
 vtss_init_conf_set, 634
 vtss_inst_create, 633
 vtss_inst_destroy, 634
 vtss_inst_get, 633
 vtss_miim_read_t, 628
 vtss_miim_write_t, 629
 vtss_mmd_read_inc_t, 630
 vtss_mmd_read_t, 629
 vtss_mmd_write_t, 630
 vtss_port_mux_mode_t, 632
 vtss_qs_conf_get, 637
 vtss_qs_conf_set, 636
 vtss_qs_mode_t, 632
 vtss_reg_read_t, 625
 vtss_reg_write_t, 626
 vtss_restart_conf_end, 635
 vtss_restart_conf_get, 636
 vtss_restart_conf_set, 636
 vtss_restart_status_get, 635
 vtss_restart_t, 633
 vtss_spi_32bit_read_write_t, 627
 vtss_spi_64bit_read_write_t, 628
 vtss_spi_read_write_t, 627
 vtss_target_type_t, 631
vtss_init_conf_get
 vtss_init_api.h, 634
vtss_init_conf_set
 vtss_init_api.h, 634
vtss_init_conf_t, 141
 miim_read, 142
 miim_write, 142
 mmd_read, 142
 mmd_read_inc, 143
 mmd_write, 143
 mux_mode, 144
 pi, 144
 qs_conf, 145
 reg_read, 142
 reg_write, 142
 restart_info_port, 144
 restart_info_src, 144
 serdes, 145
 spi_32bit_read_write, 143
 spi_64bit_read_write, 143
 spi_read_write, 143
 warm_start_enable, 144
vtss_inst_create
 vtss_init_api.h, 633
vtss_inst_create_t, 145
 target, 146
vtss_inst_destroy
 vtss_init_api.h, 634
vtss_inst_get
 vtss_init_api.h, 633
vtss_internal_bw_t
 vtss_port_api.h, 1035
vtss_intr_cfg
 vtss_misc_api.h, 739
vtss_intr_sticky_clear
 vtss_misc_api.h, 731
vtss_ip_addr_t, 146
 addr, 147
 ipv4, 146
 ipv6, 147
 type, 146
vtss_ip_network_t, 147
 address, 148
 prefix_size, 148
vtss_ip_type_t
 types.h, 606
vtss_ipv4_mc_flood_members_get
 vtss_l2_api.h, 694
vtss_ipv4_mc_flood_members_set
 vtss_l2_api.h, 694
vtss_ipv4_network_t, 148
 address, 148
 prefix_size, 149
vtss_ipv4_uc_t, 149
 destination, 150
 network, 149
vtss_ipv6_mc_ctrl_flood_get
 vtss_l2_api.h, 695
vtss_ipv6_mc_ctrl_flood_set
 vtss_l2_api.h, 696
vtss_ipv6_mc_flood_members_get
 vtss_l2_api.h, 694
vtss_ipv6_mc_flood_members_set
 vtss_l2_api.h, 695
vtss_ipv6_network_t, 150
 address, 150
 prefix_size, 151
vtss_ipv6_t, 151

addr, 151
vtss_ipv6_uc_t, 152
 destination, 152
 network, 152
vtss_irq_conf_get
 vtss_misc_api.h, 739
vtss_irq_conf_set
 vtss_misc_api.h, 740
vtss_irq_conf_t, 153
 destination, 153
 external, 153
vtss_irq_enable
 vtss_misc_api.h, 741
vtss_irq_status_get_and_mask
 vtss_misc_api.h, 740
vtss_irq_status_t, 154
 active, 154
 raw_ident, 154
 raw_mask, 155
 raw_status, 155
vtss_irq_t
 vtss_misc_api.h, 724
vtss_isidx_t
 types.h, 600
vtss_isolated_port_members_get
 vtss_l2_api.h, 679
vtss_isolated_port_members_set
 vtss_l2_api.h, 679
vtss_isolated_vlan_get
 vtss_l2_api.h, 678
vtss_isolated_vlan_set
 vtss_l2_api.h, 678
vtss_l2_api.h
 VTSS_ERPI_ARRAY_SIZE, 653
 VTSS_ERPI_END, 653
 VTSS_ERPI_START, 653
 VTSS_ERPIS, 652
 VTSS_MAC_ADDRS, 646
 VTSS_MSTI_ARRAY_SIZE, 648
 VTSS_MSTI_END, 648
 VTSS_MSTI_START, 647
 VTSS_MSTIS, 647
 VTSS_PVLAN_ARRAY_SIZE, 652
 VTSS_PVLAN_NO_DEFAULT, 652
 VTSS_PVLAN_NO_END, 652
 VTSS_PVLAN_NO_START, 652
 VTSS_PVLANS, 651
 VTSS_UPSPN_CPU, 647
 VTSS_UPSPN_NONE, 647
 VTSS_VCE_ID_LAST, 649
 VTSS_VCL_ARRAY_SIZE, 649
 VTSS_VCL_ID_END, 648
 VTSS_VCL_ID_START, 648
 VTSS_VCL_IDS, 648
 VTSS_VLAN_TRANS_FIRST_GROUP_ID, 650
 VTSS_VLAN_TRANS_GROUP_MAX_CNT, 649
 VTSS_VLAN_TRANS_LAST_GROUP_ID, 650
 VTSS_VLAN_TRANS_MAX_CNT, 649
VTSS_VLAN_TRANS_MAX_VLAN_ID, 650
VTSS_VLAN_TRANS_NULL_CHECK, 651
VTSS_VLAN_TRANS_NULL_GROUP_ID, 649
VTSS_VLAN_TRANS_PORT_BF_SIZE, 651
VTSS_VLAN_TRANS_VALID_GROUP_CHECK,
 650
VTSS_VLAN_TRANS_VALID_VLAN_CHECK,
 651
VTSS_VLAN_TRANS_VID_START, 650
VTSS_VSTAX_UPSID_LEGAL, 646
VTSS_VSTAX_UPSID_MAX, 646
VTSS_VSTAX_UPSID_MIN, 646
VTSS_VSTAX_UPSID_START, 646
VTSS_VSTAX_UPSID_UNDEF, 647
VTSS_VSTAX_UPSIDS, 646
vtss_aggr_glag_members_get, 685
vtss_aggr_mode_get, 685
vtss_aggr_mode_set, 685
vtss_aggr_port_members_get, 684
vtss_aggr_port_members_set, 684
vtss_apvlan_port_members_get, 681
vtss_apvlan_port_members_set, 681
vtss_dgroup_port_conf_get, 681
vtss_dgroup_port_conf_set, 682
vtss_eps_port_conf_get, 696
vtss_eps_port_conf_set, 696
vtss_eps_port_selector_get, 697
vtss_eps_port_selector_set, 697
vtss_eps_port_type_t, 655
vtss_eps_selector_t, 655
vtss_erps_port_state_get, 699
vtss_erps_port_state_set, 699
vtss_erps_state_t, 656
vtss_erps_vlan_member_get, 698
vtss_erps_vlan_member_set, 698
vtss_ipv4_mc_flood_members_get, 694
vtss_ipv4_mc_flood_members_set, 694
vtss_ipv6_mc_ctrl_flood_get, 695
vtss_ipv6_mc_ctrl_flood_set, 696
vtss_ipv6_mc_flood_members_get, 694
vtss_ipv6_mc_flood_members_set, 695
vtss_isolated_port_members_get, 679
vtss_isolated_port_members_set, 679
vtss_isolated_vlan_get, 678
vtss_isolated_vlan_set, 678
vtss_learn_port_mode_get, 664
vtss_learn_port_mode_set, 665
vtss_mac_table_add, 656
vtss_mac_table_age, 660
vtss_mac_table_age_time_get, 659
vtss_mac_table_age_time_set, 659
vtss_mac_table_del, 658
vtss_mac_table_flush, 660
vtss_mac_table_get, 658
vtss_mac_table_get_next, 659
vtss_mac_table_glag_add, 663
vtss_mac_table_glag_flush, 663
vtss_mac_table_port_flush, 661

vtss_mac_table_status_get, 664
vtss_mac_table_upsid_flush, 662
vtss_mac_table_upsid_upspn_flush, 662
vtss_mac_table_vlan_age, 660
vtss_mac_table_vlan_flush, 661
vtss_mac_table_vlan_glag_flush, 664
vtss_mac_table_vlan_port_flush, 662
vtss_mc_flood_members_get, 693
vtss_mc_flood_members_set, 693
vtss_mirror_conf_get, 688
vtss_mirror_conf_set, 688
vtss_mirror_cpu_egress_get, 691
vtss_mirror_cpu_egress_set, 692
vtss_mirror_cpu_ingress_get, 691
vtss_mirror_cpu_ingress_set, 691
vtss_mirror_egress_ports_get, 690
vtss_mirror_egress_ports_set, 690
vtss_mirror_ingress_ports_get, 689
vtss_mirror_ingress_ports_set, 689
vtss_mirror_monitor_port_get, 688
vtss_mirror_monitor_port_set, 689
vtss_mirror_tag_t, 655
vtss_mstp_port_msti_state_get, 668
vtss_mstp_port_msti_state_set, 668
vtss_mstp_vlan_msti_get, 667
vtss_mstp_vlan_msti_set, 667
vtss_port_state_get, 665
vtss_port_state_set, 666
vtss_pvlan_port_members_get, 680
vtss_pvlan_port_members_set, 680
vtss_sfflow_port_conf_get, 682
vtss_sfflow_port_conf_set, 683
vtss_sfflow_sampling_rate_convert, 683
vtss_stp_port_state_get, 666
vtss_stp_port_state_set, 666
vtss_stp_state_t, 653
vtss_uc_flood_members_get, 692
vtss_uc_flood_members_set, 693
vtss_vce_add, 675
vtss_vce_del, 675
vtss_vce_init, 675
vtss_vce_type_t, 654
vtss_vcl_port_conf_get, 674
vtss_vcl_port_conf_set, 674
vtss_vlan_conf_get, 669
vtss_vlan_conf_set, 669
vtss_vlan_port_conf_get, 669
vtss_vlan_port_conf_set, 671
vtss_vlan_port_members_get, 671
vtss_vlan_port_members_set, 672
vtss_vlan_port_type_t, 654
vtss_vlan_trans_group_add, 676
vtss_vlan_trans_group_del, 676
vtss_vlan_trans_group_get, 677
vtss_vlan_trans_group_to_port_get, 678
vtss_vlan_trans_group_to_port_set, 677
vtss_vlan_tx_tag_get, 673
vtss_vlan_tx_tag_set, 673
vtss_vlan_tx_tag_t, 654
vtss_vlan_vid_conf_get, 672
vtss_vlan_vid_conf_set, 672
vtss_vstax_conf_get, 699
vtss_vstax_conf_set, 700
vtss_vstax_glag_get, 687
vtss_vstax_glag_set, 687
vtss_vstax_master_upsid_get, 701
vtss_vstax_master_upsid_set, 702
vtss_vstax_port_conf_get, 700
vtss_vstax_port_conf_set, 701
vtss_vstax_topology_set, 702
vtss_vstax_topology_type_t, 656
vtss_l3_id_t, 653
vtss_l3_api.h
 VTSS_ARP_CNT, 705
 VTSS_ARP_IPV4_RELATIONS, 706
 VTSS_ARP_IPV6_RELATIONS, 706
 VTSS_JR1_ARP_CNT, 705
 VTSS_JR1_LPM_CNT, 704
 VTSS_JR1_RLEG_CNT, 705
 VTSS_LPM_CNT, 705
 VTSS_RLEG_CNT, 705
 vtss_l3_common_get, 707
 vtss_l3_common_set, 708
 vtss_l3_counters_reset, 712
 vtss_l3_counters_rleg_clear, 713
 vtss_l3_counters_rleg_get, 713
 vtss_l3_counters_system_get, 713
 vtss_l3_flush, 707
 vtss_l3_neighbour_add, 712
 vtss_l3_neighbour_del, 712
 vtss_l3_neighbour_get, 711
 vtss_l3_neighbour_type_t, 706
 vtss_l3_rleg_add, 709
 vtss_l3_rleg_common_mode_t, 706
 vtss_l3_rleg_del, 710
 vtss_l3_rleg_get, 708
 vtss_l3_rleg_get_specific, 708
 vtss_l3_rleg_update, 709
 vtss_l3_route_add, 710
 vtss_l3_route_del, 711
 vtss_l3_route_get, 710
vtss_l3_common_conf_t, 155
 base_address, 156
 rleg_mode, 156
 routing_enable, 156
vtss_l3_common_get
 vtss_l3_api.h, 707
vtss_l3_common_set
 vtss_l3_api.h, 708
vtss_l3_counters_reset
 vtss_l3_api.h, 712
vtss_l3_counters_rleg_clear
 vtss_l3_api.h, 713
vtss_l3_counters_rleg_get
 vtss_l3_api.h, 713
vtss_l3_counters_system_get

vtss_l3_api.h, 713
 vtss_l3_counters_t, 156
 ipv4uc_received_frames, 157
 ipv4uc_received_octets, 157
 ipv4uc_transmitted_frames, 158
 ipv4uc_transmitted_octets, 157
 ipv6uc_received_frames, 157
 ipv6uc_received_octets, 157
 ipv6uc_transmitted_frames, 158
 ipv6uc_transmitted_octets, 158
 vtss_l3_flush
 vtss_l3_api.h, 707
 vtss_l3_neighbour_add
 vtss_l3_api.h, 712
 vtss_l3_neighbour_del
 vtss_l3_api.h, 712
 vtss_l3_neighbour_get
 vtss_l3_api.h, 711
 vtss_l3_neighbour_t, 158
 dip, 159
 dmac, 159
 vlan, 159
 vtss_l3_neighbour_type_t
 vtss_l3_api.h, 706
 vtss_l3_rleg_add
 vtss_l3_api.h, 709
 vtss_l3_rleg_common_mode_t
 vtss_l3_api.h, 706
 vtss_l3_rleg_conf_t, 160
 ipv4_icmp_redirect_enable, 160
 ipv4_unicast_enable, 160
 ipv6_icmp_redirect_enable, 161
 ipv6_unicast_enable, 160
 vlan, 161
 vrid0, 161
 vrid0_enable, 161
 vrid1, 162
 vrid1_enable, 162
 vtss_l3_rleg_del
 vtss_l3_api.h, 710
 vtss_l3_rleg_get
 vtss_l3_api.h, 708
 vtss_l3_rleg_get_specific
 vtss_l3_api.h, 708
 vtss_l3_rleg_update
 vtss_l3_api.h, 709
 vtss_l3_route_add
 vtss_l3_api.h, 710
 vtss_l3_route_del
 vtss_l3_api.h, 711
 vtss_l3_route_get
 vtss_l3_api.h, 710
 vtss_lb_type_t
 vtss_phy_10g_api.h, 834
 vtss_lcpll_status_t, 162
 cal_done, 163
 cal_error, 163
 fsm_lock, 163
 fsm_stat, 163
 gain_stat, 164
 lock_status, 163
 vtss_learn_mode_t, 164
 automatic, 164
 cpu, 165
 discard, 165
 vtss_learn_port_mode_get
 vtss_l2_api.h, 664
 vtss_learn_port_mode_set
 vtss_l2_api.h, 665
 vtss_mac_addr_t
 types.h, 599
 vtss_mac_t, 165
 addr, 166
 vtss_mac_table_add
 vtss_l2_api.h, 656
 vtss_mac_table_age
 vtss_l2_api.h, 660
 vtss_mac_table_age_time_get
 vtss_l2_api.h, 659
 vtss_mac_table_age_time_set
 vtss_l2_api.h, 659
 vtss_mac_table_del
 vtss_l2_api.h, 658
 vtss_mac_table_entry_t, 166
 aged, 167
 copy_to_cpu, 167
 cpu_queue, 167
 destination, 167
 enable, 168
 locked, 167
 remote_entry, 168
 upsid, 168
 upspn, 168
 vid_mac, 166
 vstax2, 168
 vtss_mac_table_flush
 vtss_l2_api.h, 660
 vtss_mac_table_get
 vtss_l2_api.h, 658
 vtss_mac_table_get_next
 vtss_l2_api.h, 659
 vtss_mac_table_glag_add
 vtss_l2_api.h, 663
 vtss_mac_table_glag_flush
 vtss_l2_api.h, 663
 vtss_mac_table_port_flush
 vtss_l2_api.h, 661
 vtss_mac_table_status_get
 vtss_l2_api.h, 664
 vtss_mac_table_status_t, 169
 aged, 170
 learned, 169
 moved, 170
 replaced, 169
 vtss_mac_table_upsid_flush
 vtss_l2_api.h, 662

vtss_mac_table_upsid_upspn_flush
 vtss_l2_api.h, 662

vtss_mac_table_vlan_age
 vtss_l2_api.h, 660

vtss_mac_table_vlan_flush
 vtss_l2_api.h, 661

vtss_mac_table_vlan_glag_flush
 vtss_l2_api.h, 664

vtss_mac_table_vlan_port_flush
 vtss_l2_api.h, 662

vtss_mc_flood_members_get
 vtss_l2_api.h, 693

vtss_mc_flood_members_set
 vtss_l2_api.h, 693

vtss_mem_flags_t
 types.h, 603

vtss_miim_controller_t
 vtss_port_api.h, 1035

vtss_miim_read
 vtss_port_api.h, 1044

vtss_miim_read_t
 vtss_init_api.h, 628

vtss_miim_write
 vtss_port_api.h, 1044

vtss_miim_write_t
 vtss_init_api.h, 629

vtss_mirror_conf_get
 vtss_l2_api.h, 688

vtss_mirror_conf_set
 vtss_l2_api.h, 688

vtss_mirror_conf_t, 170
 dei, 172
 fwd_enable, 171
 pcp, 171
 port_no, 171
 tag, 171
 vid, 171

vtss_mirror_cpu_egress_get
 vtss_l2_api.h, 691

vtss_mirror_cpu_egress_set
 vtss_l2_api.h, 692

vtss_mirror_cpu_ingress_get
 vtss_l2_api.h, 691

vtss_mirror_cpu_ingress_set
 vtss_l2_api.h, 691

vtss_mirror_egress_ports_get
 vtss_l2_api.h, 690

vtss_mirror_egress_ports_set
 vtss_l2_api.h, 690

vtss_mirror_ingress_ports_get
 vtss_l2_api.h, 689

vtss_mirror_ingress_ports_set
 vtss_l2_api.h, 689

vtss_mirror_monitor_port_get
 vtss_l2_api.h, 688

vtss_mirror_monitor_port_set
 vtss_l2_api.h, 689

vtss_mirror_tag_t

vtss_l2_api.h, 655

vtss_misc_api.h
 tod_get_ns_cnt_cb_t, 720
 VTSS_OS_TIMESTAMP_TYPE, 720
 VTSS_OS_TIMESTAMP, 720

vtss_callout_lock, 728

vtss_callout_trace_hex_dump, 727

vtss_callout_trace_printf, 726

vtss_callout_unlock, 728

vtss_chip_id_get, 731

vtss_debug_group_t, 722

vtss_debug_info_get, 727

vtss_debug_info_print, 728

vtss_debug_layer_t, 722

vtss_debug_lock, 729

vtss_debug_reg_check_set, 745

vtss_debug_unlock, 729

vtss_dev_all_event_enable, 733

vtss_dev_all_event_poll, 733

vtss_eee_port_conf_set, 744

vtss_eee_port_counter_get, 745

vtss_eee_port_state_set, 744

vtss_eee_state_select_t, 725

vtss_fan_controller_init, 743

vtss_fan_cool_lvl_get, 744

vtss_fan_cool_lvl_set, 743

vtss_fan_rotation_get, 742

vtss_gpio_direction_set, 734

vtss_gpio_event_enable, 736

vtss_gpio_event_poll, 736

vtss_gpio_mode_set, 734

vtss_gpio_mode_t, 723

vtss_gpio_read, 735

vtss_gpio_write, 735

vtss_intr_cfg, 739

vtss_intr_sticky_clear, 731

vtss_irq_conf_get, 739

vtss_irq_conf_set, 740

vtss_irq_enable, 741

vtss_irq_status_get_and_mask, 740

vtss_irq_t, 724

vtss_poll_1sec, 732

vtss_ptp_event_enable, 732

vtss_ptp_event_poll, 732

vtss_reg_read, 729

vtss_reg_write, 730

vtss_reg_write_masked, 730

vtss_sgpiobmode_t, 724

vtss_sgpiocfg_get, 737

vtss_sgpiocfg_set, 737

vtss_sgpioevent_enable, 738

vtss_sgpioeventpoll, 738

vtss_sgpiemode_t, 724

vtss_sgpioread, 737

vtss_temp_sensor_get, 742

vtss_temp_sensor_init, 742

vtss_tod_get_ns_cnt, 741

vtss_tod_set_ns_cnt_cb, 741

vtss_trace_conf_get, 725
 vtss_trace_conf_set, 726
 vtss_trace_group_t, 721
 vtss_trace_layer_t, 720
 vtss_trace_level_t, 721
 vtss_mmd_read
 vtss_port_api.h, 1047
 vtss_mmd_read_inc_t
 vtss_init_api.h, 630
 vtss_mmd_read_t
 vtss_init_api.h, 629
 vtss_mmd_write
 vtss_port_api.h, 1047
 vtss_mmd_write_t
 vtss_init_api.h, 630
 vtss_mstp_port_msti_state_get
 vtss_l2_api.h, 668
 vtss_mstp_port_msti_state_set
 vtss_l2_api.h, 668
 vtss_mstp_vlan_msti_get
 vtss_l2_api.h, 667
 vtss_mstp_vlan_msti_set
 vtss_l2_api.h, 667
 vtss_mtimmer_t, 172
 now, 173
 timeout, 172
 vtss_os_custom.h, 752
 vtss_os_ecos.h, 760
 vtss_npi_conf_get
 vtss_packet_api.h, 779
 vtss_npi_conf_set
 vtss_packet_api.h, 779
 vtss_npi_conf_t, 173
 enable, 173
 port_no, 174
 vtss_os_custom.h
 uint, 748
 ulong, 748
 VTSS_DIV64, 749
 VTSS_LABS, 750
 VTSS_LLabs, 750
 VTSS_MOD64, 749
 VTSS_MSLEEP, 748
 VTSS_MTIMER_CANCEL, 749
 VTSS_MTIMER_START, 749
 VTSS_MTIMER_TIMEOUT, 749
 VTSS_OS_CTZ64, 750
 VTSS_OS_CTZ, 750
 VTSS_OS_FREE, 751
 VTSS_OS_MALLOC, 751
 VTSS_OS_RAND, 751
 vtss_mtimmer_t, 752
 vtss_os_ecos.h
 llabs, 761
 VTSS_DIV64, 754
 VTSS_LABS, 755
 VTSS_LLabs, 755
 VTSS_MOD64, 755
 VTSS_MSLEEP, 753
 VTSS_MTIMER_CANCEL, 754
 VTSS_MTIMER_START, 754
 VTSS_MTIMER_TIMEOUT, 754
 VTSS_NSLEEP, 753
 VTSS_OS_BIG_ENDIAN, 758
 VTSS_OS_COMPILER_ATTRIBUTE_ALIGNED, 757
 VTSS_OS_CTZ64, 756
 VTSS_OS_CTZ, 755
 VTSS_OS_DCACHE_FLUSH, 758
 VTSS_OS_DCACHE_INVALIDATE, 758
 VTSS_OS_DCACHE_LINE_SIZE_BYTES, 757
 VTSS_OS_FREE, 756
 VTSS_OS_INTERRUPT_DISABLE, 760
 VTSS_OS_INTERRUPT_FLAGS, 760
 VTSS_OS_INTERRUPT_RESTORE, 760
 VTSS_OS_MALLOC, 756
 VTSS_OS_NTOHL, 759
 VTSS_OS_RAND, 757
 VTSS_OS_REORDER_BARRIER, 757
 VTSS_OS_SCHEDULER_FLAGS, 759
 VTSS_OS_SCHEDULER_LOCK, 759
 VTSS_OS_SCHEDULER_UNLOCK, 759
 VTSS_OS_VIRT_TO_PHYS, 758
 VTSS_TIME_OF_DAY, 754
 vtss_callout_free, 761
 vtss_callout_malloc, 761
 vtss_mtimmer_t, 760
 vtss_os_linux.h
 VTSS_DIV64, 766
 VTSS_LABS, 766
 VTSS_LLabs, 767
 VTSS_MOD64, 766
 VTSS_MSLEEP, 764
 VTSS_MTIMER_CANCEL, 765
 VTSS_MTIMER_START, 764
 VTSS_MTIMER_TIMEOUT, 764
 VTSS_NSLEEP, 763
 VTSS_OS_BIG_ENDIAN, 763
 VTSS_OS_CTZ64, 767
 VTSS_OS_CTZ, 767
 VTSS_OS_FREE, 768
 VTSS_OS_MALLOC, 768
 VTSS_OS_NTOHL, 763
 VTSS_OS_RAND, 768
 VTSS_OS_SCHEDULER_FLAGS, 765
 VTSS_OS_SCHEDULER_LOCK, 765
 VTSS_OS_SCHEDULER_UNLOCK, 766
 VTSS_TIME_OF_DAY, 765
 vtss_os_timestamp_t, 174
 hw_cnt, 174
 vtss_packet_api.h
 VTSS_JR1_PACKET_HDR_SIZE_BYTES, 773
 VTSS_JR1_RX_IFH_SIZE, 774
 VTSS_JR2_PACKET_HDR_SIZE_BYTES, 773
 VTSS_JR2_RX_IFH_SIZE, 775
 VTSS_L26_PACKET_HDR_SIZE_BYTES, 774

VTSS_L26_RX_IFH_SIZE, 775
VTSS_PACKET_HDR_SIZE_BYTES, 774
VTSS_PACKET_TX_IFH_MAX, 775
VTSS_PRIO_SUPER, 773
VTSS_SVL_PACKET_HDR_SIZE_BYTES, 774
VTSS_SVL_RX_IFH_SIZE, 774
VTSS_VSTAX_HDR_SIZE, 773
VTSS_VSTAX_TTL_PORT, 773
vtss_npi_conf_get, 779
vtss_npi_conf_set, 779
vtss_packet_dma_conf_get, 791
vtss_packet_dma_conf_set, 791
vtss_packet_dma_offset, 791
vtss_packet_filter_t, 775
vtss_packet_frame_filter, 785
vtss_packet_frame_info_init, 785
vtss_packet_oam_type_t, 776
vtss_packet_port_filter_get, 786
vtss_packet_port_info_init, 786
vtss_packet_ptp_action_t, 777
vtss_packet_rx_conf_get, 780
vtss_packet_rx_conf_set, 780
vtss_packet_rx_frame_discard, 783
vtss_packet_rx_frame_get, 782
vtss_packet_rx_frame_get_raw, 783
vtss_packet_rx_hdr_decode, 788
vtss_packet_rx_hints_t, 777
vtss_packet_rx_port_conf_get, 780
vtss_packet_rx_port_conf_set, 782
vtss_packet_tx_frame, 790
vtss_packet_tx_frame_port, 784
vtss_packet_tx_frame_port_vlan, 784
vtss_packet_tx_frame_vlan, 785
vtss_packet_tx_frame_vstax, 787
vtss_packet_tx_hdr_compile, 789
vtss_packet_tx_hdr_encode, 788
vtss_packet_tx_info_init, 790
vtss_packet_tx_vstax_t, 778
vtss_packet_vstax_frame2header, 787
vtss_packet_vstax_header2frame, 787
vtss_tag_type_t, 777
vtss_vstax_fwd_mode_t, 776
vtss_packet_dma_conf_get
 vtss_packet_api.h, 791
vtss_packet_dma_conf_set
 vtss_packet_api.h, 791
vtss_packet_dma_conf_t, 175
 dma_enable, 175
vtss_packet_dma_offset
 vtss_packet_api.h, 791
vtss_packet_filter_t
 vtss_packet_api.h, 775
vtss_packet_frame_filter
 vtss_packet_api.h, 785
vtss_packet_frame_info_init
 vtss_packet_api.h, 785
vtss_packet_frame_info_t, 176
 aggr_rx_disable, 177
 aggr_tx_disable, 177
 glag_no, 176
 port_no, 176
 port_tx, 177
 vid, 176
 vtss_packet_oam_type_t
 vtss_packet_api.h, 776
 vtss_packet_port_filter_get
 vtss_packet_api.h, 786
 vtss_packet_port_filter_t, 177
 filter, 178
 tpid, 178
vtss_packet_port_info_init
 vtss_packet_api.h, 786
vtss_packet_port_info_t, 178
 aggr_rx_disable, 179
 aggr_tx_disable, 180
 glag_no, 179
 port_no, 179
 vid, 179
vtss_packet_ptp_action_t
 vtss_packet_api.h, 777
vtss_packet_reg_type_t
 types.h, 605
vtss_packet_rx_conf_get
 vtss_packet_api.h, 780
vtss_packet_rx_conf_set
 vtss_packet_api.h, 780
vtss_packet_rx_conf_t, 180
 grp_map, 181
 map, 181
 queue, 180
 reg, 181
vtss_packet_rx_frame_discard
 vtss_packet_api.h, 783
vtss_packet_rx_frame_get
 vtss_packet_api.h, 782
vtss_packet_rx_frame_get_raw
 vtss_packet_api.h, 783
vtss_packet_rx_grp_t
 types.h, 600
vtss_packet_rx_hdr_decode
 vtss_packet_api.h, 788
vtss_packet_rx_header_t, 181
 arrived_tagged, 183
 learn, 182
 length, 182
 port_no, 182
 queue_mask, 182
 tag, 183
 vstax, 183
vtss_packet_rx_hints_t
 vtss_packet_api.h, 777
vtss_packet_rx_info_t, 183
 acl_hit, 188
 acl_idx, 188
 cos, 188
 glag_no, 185

hints, 184
 hw_tstamp, 190
 hw_tstamp_decoded, 190
 isdx, 192
 length, 185
 oam_info, 191
 oam_info_decoded, 192
 port_no, 185
 sflow_port_no, 191
 sflow_type, 190
 stripped_tag, 187
 sw_tstamp, 189
 tag, 186
 tag_type, 186
 tstamp_id, 189
 tstamp_id_decoded, 189
 vstax, 192
 xtr_qu_mask, 187
vtss_packet_rx_meta_t, 193
 chip_no, 194
 etype, 195
 fcs, 195
 length, 196
 no_wait, 194
 sw_tstamp, 196
 xtr_qu, 194
vtss_packet_rx_port_conf_get
 vtss_packet_api.h, 780
vtss_packet_rx_port_conf_set
 vtss_packet_api.h, 782
vtss_packet_rx_port_conf_t, 197
 bpdu_reg, 198
 garp_reg, 198
vtss_packet_rx_queue_conf_t, 198
 npi, 199
 size, 198
vtss_packet_rx_queue_map_t, 199
 bpdu_queue, 200
 garp_queue, 200
 igmp_queue, 200
 ipmc_ctrl_queue, 200
 l3_other_queue, 201
 l3_uc_queue, 201
 learn_queue, 200
 lrn_all_queue, 201
 mac_vid_queue, 200
 sflow_queue, 201
 stack_queue, 201
vtss_packet_rx_queue_npi_conf_t, 202
 enable, 202
vtss_packet_rx_reg_t, 203
 bpdu_cpu_only, 203
 garp_cpu_only, 203
 igmp_cpu_only, 204
 ipmc_ctrl_cpu_copy, 203
 mld_cpu_only, 204
vtss_packet_tx_frame
 vtss_packet_api.h, 790

vtss_packet_tx_frame_port
 vtss_packet_api.h, 784
vtss_packet_tx_frame_port_vlan
 vtss_packet_api.h, 784
vtss_packet_tx_frame_vlan
 vtss_packet_api.h, 785
vtss_packet_tx_frame_vstax
 vtss_packet_api.h, 787
vtss_packet_tx_grp_t
 types.h, 600
vtss_packet_tx_hdr_compile
 vtss_packet_api.h, 789
vtss_packet_tx_hdr_encode
 vtss_packet_api.h, 788
vtss_packet_tx_ifh_t, 204
 ifh, 205
 length, 205
vtss_packet_tx_info_init
 vtss_packet_api.h, 790
vtss_packet_tx_info_t, 205
 agrr_code, 208
 bin, 210
 cos, 209
 dp, 214
 dst_port_mask, 206
 frm_len, 207
 isdx, 213
 isdx_dont_use, 214
 latch_timestamp, 212
 masquerade_port, 215
 oam_type, 213
 pdu_offset, 215
 ptp_action, 211
 ptp_id, 211
 ptp_timestamp, 212
 switch_frm, 206
 sym, 210
 tag, 207
 tx_vstax_hdr, 209
 vstax, 211
vtss_packet_tx_vstax_t
 vtss_packet_api.h, 778
vtss_packet_vstax_frame2header
 vtss_packet_api.h, 787
vtss_packet_vstax_header2frame
 vtss_packet_api.h, 787
vtss_phy_10G_is_valid
 vtss_phy_10g_api.h, 866
vtss_phy_10g_apc_conf_get
 vtss_phy_10g_api.h, 846
vtss_phy_10g_apc_conf_set
 vtss_phy_10g_api.h, 846
vtss_phy_10g_apc_conf_t, 216
 op_mode, 216
 op_mode_flag, 217
vtss_phy_10g_apc_restart
 vtss_phy_10g_api.h, 847
vtss_phy_10g_apc_status_get

vtss_phy_10g_api.h, 847
vtss_phy_10g_apc_status_t, 217
freeze, 218
reset, 217
vtss_phy_10g_api.h
AMPLITUDE_POINTS, 808
apc_ib_regulator_t, 825
BOOLEAN_STORAGE_COUNT, 808
ckout_sel_, 832
ckout_sel_t, 821
clk_mstr_t, 826
ddr_mode_t, 825
oper_mode_t, 822
PHASE_POINTS, 808
UNSIGNED_STORAGE_COUNT, 808
VTSS_10G_PHY_GPIO_MAL_MAX, 809
VTSS_10G_PHY_GPIO_MAX, 809
VTSS_PHY_10G_FEC_FIXED_CNT_THRESH←
_EV, 818
VTSS_PHY_10G_FEC_UNFIXED_CNT_THRE←
SH_EV, 818
VTSS_PHY_10G_GPIO_INT_AGG0_EV, 819
VTSS_PHY_10G_GPIO_INT_AGG1_EV, 819
VTSS_PHY_10G_GPIO_INT_AGG2_EV, 819
VTSS_PHY_10G_GPIO_INT_AGG3_EV, 819
VTSS_PHY_10G_HIGH_BER_EV, 810
VTSS_PHY_10G_HIGHBER_EV, 818
VTSS_PHY_10G_HOST_MAC_LOCAL_FAULT←
_EV, 820
VTSS_PHY_10G_HOST_MAC_REMOTE_FAU←
LT_EV, 821
VTSS_PHY_10G_LINE_MAC_LOCAL_FAULT←
EV, 820
VTSS_PHY_10G_LINE_MAC_REMOTE_FAUL←
T_EV, 820
VTSS_PHY_10G_LINK_LOS_EV, 810
VTSS_PHY_10G_LOPC_EV, 810
VTSS_PHY_10G_MACSEC_DISABLED, 809
VTSS_PHY_10G_MACSEC_KEY_128, 809
VTSS_PHY_10G_MODULE_STAT_EV, 811
VTSS_PHY_10G_ONE_LINE_ACTIVE, 808
VTSS_PHY_10G_PCS_RECEIVE_FAULT_EV,
811
VTSS_PHY_10G_RX_BLK_DEC_CNT_THRES←
H_EV, 817
VTSS_PHY_10G_RX_CHAR_DEC_CNT_THR←
ESH_EV, 817
VTSS_PHY_10G_RX_LINK_STAT_EV, 818, 819
VTSS_PHY_10G_RX_LOL_EV, 810, 816
VTSS_PHY_10G_RX_LOS_EV, 816
VTSS_PHY_10G_RX_SEQ_CNT_THRESH_EV,
817
VTSS_PHY_10G_TIMESTAMP_DISABLED, 809
VTSS_PHY_10G_TX_BLK_ENC_CNT_THRES←
H_EV, 817
VTSS_PHY_10G_TX_CHAR_ENC_CNT_THRE←
SH_EV, 817
VTSS_PHY_10G_TX_LOL_EV, 810, 816
VTSS_PHY_10G_TX_SEQ_CNT_THRESH_EV,
818
VTSS_PHY_10G_HOST_AUTONEG_RESTART←
_EV, 820
VTSS_PHY_1G_LINE_AUTONEG_RESTART←
EV, 820
VTSS_PHY_EWIS_AISL_EV, 812
VTSS_PHY_EWIS_AISP_EV, 813
VTSS_PHY_EWIS_B1_NZ_EV, 814
VTSS_PHY_EWIS_B1_THRESH_EV, 815
VTSS_PHY_EWIS_B2_NZ_EV, 814
VTSS_PHY_EWIS_B2_THRESH_EV, 815
VTSS_PHY_EWIS_B3_NZ_EV, 814
VTSS_PHY_EWIS_B3_THRESH_EV, 815
VTSS_PHY_EWIS_FAIS_EV, 811
VTSS_PHY_EWIS_FERDIP_EV, 813
VTSS_PHY_EWIS_FEUNEQP_EV, 813
VTSS_PHY_EWIS_FPLM_EV, 811
VTSS_PHY_EWIS_LCDP_EV, 812
VTSS_PHY_EWIS_LOF_EV, 812
VTSS_PHY_EWIS_LOPP_EV, 813
VTSS_PHY_EWIS_PLMP_EV, 812
VTSS_PHY_EWIS_RDIL_EV, 812
VTSS_PHY_EWIS_REL_EV, 814
VTSS_PHY_EWIS_REL_NZ_EV, 815
VTSS_PHY_EWIS_REL_THRESH_EV, 816
VTSS_PHY_EWIS_REIP_EV, 814
VTSS_PHY_EWIS_REIP_NZ_EV, 815
VTSS_PHY_EWIS_REIP_THRESH_EV, 816
VTSS_PHY_EWIS_SEF_EV, 811
VTSS_PHY_EWIS_UNEQP_EV, 813
VTSS_SLEWRATE_120PS, 807
VTSS_SLEWRATE_25PS, 806
VTSS_SLEWRATE_35PS, 807
VTSS_SLEWRATE_55PS, 807
VTSS_SLEWRATE_70PS, 807
VTSS_SLEWRATE_INVALID, 807
vtss_10g_phy_gpio_t, 837
vtss_32_cntr_t, 821
vtss_channel_t, 828
vtss_ckout_data_sel_t, 829
vtss_gpio_10g_aggr_intrpt_t, 841
vtss_gpio_10g_chan_intrpt_t, 840
vtss_gpio_10g_gpio_intr_sgnl_t, 838
vtss_gpio_10g_input_t, 841
vtss_gpio_10g_no_t, 821
vtss_lb_type_t, 834
vtss_phy_10G_is_valid, 866
vtss_phy_10g_apc_conf_get, 846
vtss_phy_10g_apc_conf_set, 846
vtss_phy_10g_apc_restart, 847
vtss_phy_10g_apc_status_get, 847
vtss_phy_10g_auto_failover_event_t, 835
vtss_phy_10g_auto_failover_filter_t, 836
vtss_phy_10g_auto_failover_get, 868
vtss_phy_10g_auto_failover_set, 868
vtss_phy_10g_base_host_kr_train_aneg_set, 858
vtss_phy_10g_base_kr_conf_get, 859

vtss_phy_10g_base_kr_conf_set, 859
 vtss_phy_10g_base_kr_host_conf_get, 860
 vtss_phy_10g_base_kr_host_conf_set, 860
 vtss_phy_10g_base_kr_train_aneg_get, 857
 vtss_phy_10g_base_kr_train_aneg_set, 858
 vtss_phy_10g_ckout_conf_set, 854
 vtss_phy_10g_ckout_freq_t, 829
 vtss_phy_10g_clause_37_control_get, 863
 vtss_phy_10g_clause_37_control_set, 864
 vtss_phy_10g_clause_37_remote_fault_t, 833
 vtss_phy_10g_clause_37_status_get, 863
 vtss_phy_10g_clk_sel_t, 831
 vtss_phy_10g_cnt_get, 865
 vtss_phy_10g_csr_read, 883
 vtss_phy_10g_csr_write, 883
 vtss_phy_10g_debug_register_dump, 857
 vtss_phy_10g_edc_fw_status_get, 882
 vtss_phy_10g_event_enable_get, 879
 vtss_phy_10g_event_enable_set, 878
 vtss_phy_10g_event_poll, 880
 vtss_phy_10g_event_t, 821
 vtss_phy_10g_extended_event_enable_get, 879
 vtss_phy_10g_extended_event_enable_set, 881
 vtss_phy_10g_extended_event_poll, 881
 vtss_phy_10g_extnd_event_t, 822
 vtss_phy_10g_failover_get, 867
 vtss_phy_10g_failover_mode_t, 835
 vtss_phy_10g_failover_set, 867
 vtss_phy_10g_fc_buffer_reset, 882
 vtss_phy_10g_get_user_data, 885
 vtss_phy_10g_gpio_mode_get, 877
 vtss_phy_10g_gpio_mode_set, 877
 vtss_phy_10g_gpio_read, 878
 vtss_phy_10g_gpio_write, 878
 vtss_phy_10g_host_clk_conf_set, 855
 vtss_phy_10g_host_recvrd_clk_conf_set, 856
 vtss_phy_10g_i2c_read, 884
 vtss_phy_10g_i2c_write, 885
 vtss_phy_10g_ib_apc_op_mode_t, 827
 vtss_phy_10g_ib_conf_get, 845
 vtss_phy_10g_ib_conf_set, 844
 vtss_phy_10g_ib_status_get, 845
 vtss_phy_10g_id_get, 876
 vtss_phy_10g_init, 842
 vtss_phy_10g_jitter_conf_get, 848
 vtss_phy_10g_jitter_conf_set, 848
 vtss_phy_10g_jitter_status_get, 849
 vtss_phy_10g_kr_status_get, 861
 vtss_phy_10g_lane_sync_set, 856
 vtss_phy_10g_line_clk_conf_set, 855
 vtss_phy_10g_line_recvrd_clk_conf_set, 856
 vtss_phy_10g_loopback_get, 865
 vtss_phy_10g_loopback_set, 864
 vtss_phy_10g_media_t, 826
 vtss_phy_10g_mode_get, 842
 vtss_phy_10g_mode_set, 844
 vtss_phy_10g_ob_status_get, 861
 vtss_phy_10g_pcs_prbs_gen_conf_get, 870
 vtss_phy_10g_pcs_prbs_gen_conf_set, 870
 vtss_phy_10g_pcs_prbs_mon_conf_get, 871
 vtss_phy_10g_pcs_prbs_mon_conf_set, 871
 vtss_phy_10g_pcs_prbs_mon_status_get, 872
 vtss_phy_10g_pcs_status_get, 880
 vtss_phy_10g_pkt_gen_conf, 875
 vtss_phy_10g_pkt_mon_conf, 875
 vtss_phy_10g_pkt_mon_counters_get, 876
 vtss_phy_10g_pkt_mon_rst_t, 836
 vtss_phy_10g_poll_1sec, 881
 vtss_phy_10g_power_get, 866
 vtss_phy_10g_power_set, 866
 vtss_phy_10g_power_t, 835
 vtss_phy_10g_prbs_gen_conf, 872
 vtss_phy_10g_prbs_gen_conf_get, 873
 vtss_phy_10g_prbs_mon_conf, 873
 vtss_phy_10g_prbs_mon_conf_get, 874
 vtss_phy_10g_prbs_mon_status_get, 874
 vtss_phy_10g_recvrd_clk_sel_t, 832
 vtss_phy_10g_reset, 862
 vtss_phy_10g_rx_macro_t, 833
 vtss_phy_10g_rxckout_get, 851
 vtss_phy_10g_rxckout_set, 851
 vtss_phy_10g_sckout_conf_set, 854
 vtss_phy_10g_sckout_freq_t, 832
 vtss_phy_10g_serdes_status_get, 862
 vtss_phy_10g_sgmii_mode_set, 884
 vtss_phy_10g_squelch_src_t, 830
 vtss_phy_10g_srefclk_conf_get, 853
 vtss_phy_10g_srefclk_conf_set, 853
 vtss_phy_10g_srefclk_freq_t, 828
 vtss_phy_10g_status_get, 862
 vtss_phy_10g_sync_e_clkout_get, 849
 vtss_phy_10g_sync_e_clkout_set, 850
 vtss_phy_10g_tx_macro_t, 833
 vtss_phy_10g_txckout_get, 852
 vtss_phy_10g_txckout_set, 852
 vtss_phy_10g_vscope_conf_get, 869
 vtss_phy_10g_vscope_conf_set, 869
 vtss_phy_10g_vscope_scan_status_get, 869
 vtss_phy_10g_vscope_scan_t, 836
 vtss_phy_10g_xfp_clkout_get, 850
 vtss_phy_10g_xfp_clkout_set, 851
 vtss_phy_6g_link_partner_distance_t, 827
 vtss_phy_interface_mode, 823
 vtss_phy_warm_start_10g_failed_get, 884
 vtss_recvrd_clkout_t, 828
 vtss_recvrd_t, 824
 vtss_recvrdclk_cdr_div_t, 824
 vtss_rptr_rate_t, 826
 vtss_srefclk_div_t, 824
 vtss_wref_clk_div_t, 825
 vtss_wrefclk_t, 823
 vtss_phy_10g_auto_failover_conf_t, 218
 a_gpio, 220
 channel_id, 219
 enable, 220
 evnt, 219

filter, 220
fltr_val, 220
is_host_side, 219
port_no, 219
trig_ch_id, 219
v_gpio, 219
vtss_phy_10g_auto_failover_event_t
 vtss_phy_10g_api.h, 835
vtss_phy_10g_auto_failover_filter_t
 vtss_phy_10g_api.h, 836
vtss_phy_10g_auto_failover_get
 vtss_phy_10g_api.h, 868
vtss_phy_10g_auto_failover_set
 vtss_phy_10g_api.h, 868
vtss_phy_10g_base_host_kr_train_aneg_set
 vtss_phy_10g_api.h, 858
vtss_phy_10g_base_kr_autoneg_t, 221
 an_enable, 221
 an_reset, 221
 an_restart, 221
vtss_phy_10g_base_kr_conf_get
 vtss_phy_10g_api.h, 859
vtss_phy_10g_base_kr_conf_set
 vtss_phy_10g_api.h, 859
vtss_phy_10g_base_kr_conf_t, 222
 ampl, 223
 c0, 222
 c1, 223
 cm1, 222
 en_ob, 223
 ser_inv, 223
 slewrate, 223
vtss_phy_10g_base_kr_host_conf_get
 vtss_phy_10g_api.h, 860
vtss_phy_10g_base_kr_host_conf_set
 vtss_phy_10g_api.h, 860
vtss_phy_10g_base_kr_ld_adv_abil_t, 224
 adv_10g, 224
 adv_1g, 224
 fec_abil, 225
 fec_req, 225
vtss_phy_10g_base_kr_status_t, 225
 aneg, 226
 fec, 226
 train, 226
vtss_phy_10g_base_kr_train_aneg_get
 vtss_phy_10g_api.h, 857
vtss_phy_10g_base_kr_train_aneg_set
 vtss_phy_10g_api.h, 858
vtss_phy_10g_base_kr_train_aneg_t, 226
 autoneg, 227
 host_kr, 227
 ld_abil, 227
 line_kr, 227
 training, 227
vtss_phy_10g_base_kr_training_t, 228
 enable, 228
 ld_pre_init, 229
 trmhd_c0, 229
 trmhd_cm, 229
 trmhd_cp, 228
vtss_phy_10g_ckout_conf_set
 vtss_phy_10g_api.h, 854
vtss_phy_10g_ckout_conf_t, 229
 ckout_sel, 231
 enable, 231
 freq, 230
 mode, 230
 squelch_inv, 230
 src, 230
vtss_phy_10g_ckout_freq_t
 vtss_phy_10g_api.h, 829
vtss_phy_10g_clause_37_adv_t, 231
 acknowledge, 233
 asymmetric_pause, 232
 fdx, 232
 hdx, 232
 next_page, 233
 remote_fault, 232
 symmetric_pause, 232
vtss_phy_10g_clause_37_cmn_status_t, 233
 host, 234
 line, 234
vtss_phy_10g_clause_37_control_get
 vtss_phy_10g_api.h, 863
vtss_phy_10g_clause_37_control_set
 vtss_phy_10g_api.h, 864
vtss_phy_10g_clause_37_control_t, 234
 advertisement, 235
 enable, 235
 enable_pass_thru, 235
 host, 235
 l_h, 235
 line, 235
vtss_phy_10g_clause_37_remote_fault_t
 vtss_phy_10g_api.h, 833
vtss_phy_10g_clause_37_status_get
 vtss_phy_10g_api.h, 863
vtss_phy_10g_clause_37_status_t, 236
 autoneg, 237
 complete, 236
 link, 236
 partner_advertisement, 237
vtss_phy_10g_clk_sel_t
 vtss_phy_10g_api.h, 831
vtss_phy_10g_clk_src_t, 237
 is_high_amp, 238
vtss_phy_10g_cnt_get
 vtss_phy_10g_api.h, 865
vtss_phy_10g_cnt_t, 238
 pcs, 238
vtss_phy_10g_csr_read
 vtss_phy_10g_api.h, 883
vtss_phy_10g_csr_write
 vtss_phy_10g_api.h, 883
vtss_phy_10g_debug_register_dump

vtss_phy_10g_api.h, 857
 vtss_phy_10g_edc_fw_status_get
 vtss_phy_10g_api.h, 882
 vtss_phy_10g_event_enable_get
 vtss_phy_10g_api.h, 879
 vtss_phy_10g_event_enable_set
 vtss_phy_10g_api.h, 878
 vtss_phy_10g_event_poll
 vtss_phy_10g_api.h, 880
 vtss_phy_10g_event_t
 vtss_phy_10g_api.h, 821
 vtss_phy_10g_extended_event_enable_get
 vtss_phy_10g_api.h, 879
 vtss_phy_10g_extended_event_enable_set
 vtss_phy_10g_api.h, 881
 vtss_phy_10g_extended_event_poll
 vtss_phy_10g_api.h, 881
 vtss_phy_10g_extnd_event_t
 vtss_phy_10g_api.h, 822
 vtss_phy_10g_failover_get
 vtss_phy_10g_api.h, 867
 vtss_phy_10g_failover_mode_t
 vtss_phy_10g_api.h, 835
 vtss_phy_10g_failover_set
 vtss_phy_10g_api.h, 867
 vtss_phy_10g_fc_buffer_reset
 vtss_phy_10g_api.h, 882
 vtss_phy_10g_fifo_sync_t, 239
 bypass_in_api, 239
 pr, 239
 skip_rev_check, 239
 vtss_phy_10g_fw_status_t, 240
 edc_fw_api_load, 241
 edc_fw_chksum, 240
 edc_fw_rev, 240
 icpu_activity, 241
 vtss_phy_10g_get_user_data
 vtss_phy_10g_api.h, 885
 vtss_phy_10g_gpio_mode_get
 vtss_phy_10g_api.h, 877
 vtss_phy_10g_gpio_mode_set
 vtss_phy_10g_api.h, 877
 vtss_phy_10g_gpio_read
 vtss_phy_10g_api.h, 878
 vtss_phy_10g_gpio_write
 vtss_phy_10g_api.h, 878
 vtss_phy_10g_host_clk_conf_set
 vtss_phy_10g_api.h, 855
 vtss_phy_10g_host_clk_conf_t, 241
 clk_sel_no, 242
 mode, 242
 recvrd_clk_sel, 242
 vtss_phy_10g_host_recvrd_clk_conf_set
 vtss_phy_10g_api.h, 856
 vtss_phy_10g_i2c_read
 vtss_phy_10g_api.h, 884
 vtss_phy_10g_i2c_write
 vtss_phy_10g_api.h, 885
 vtss_phy_10g_ib_apc_op_mode_t
 vtss_phy_10g_api.h, 827
 vtss_phy_10g_ib_conf_get
 vtss_phy_10g_api.h, 845
 vtss_phy_10g_ib_conf_set
 vtss_phy_10g_api.h, 844
 vtss_phy_10g_ib_conf_t, 242
 agc, 244
 apc_bit_mask, 246
 c, 244
 config_bit_mask, 246
 dfe1, 244
 dfe2, 244
 dfe3, 245
 dfe4, 245
 freeze_bit_mask, 246
 gain, 243
 gainadj, 243
 is_host, 246
 l, 244
 ld, 245
 offs, 243
 prbs, 245
 prbs_inv, 245
 vtss_phy_10g_ib_status_get
 vtss_phy_10g_api.h, 845
 vtss_phy_10g_ib_status_t, 247
 bit_errors, 247
 ib_conf, 247
 sig_det, 247
 vtss_phy_10g_ib_storage_t, 248
 ib_storage, 248
 ib_storage_bool, 248
 vtss_phy_10g_id_get
 vtss_phy_10g_api.h, 876
 vtss_phy_10g_id_t, 249
 channel_id, 250
 device_feature_status, 250
 family, 250
 part_number, 249
 phy_api_base_no, 250
 revision, 249
 type, 250
 vtss_phy_10g_init
 vtss_phy_10g_api.h, 842
 vtss_phy_10g_init_parm_t, 251
 channel_conf, 251
 vtss_phy_10g_jitter_conf_get
 vtss_phy_10g_api.h, 848
 vtss_phy_10g_jitter_conf_set
 vtss_phy_10g_api.h, 848
 vtss_phy_10g_jitter_conf_t, 252
 incr_levn, 252
 levn, 252
 vtail, 252
 vtss_phy_10g_jitter_status_get
 vtss_phy_10g_api.h, 849
 vtss_phy_10g_kr_status_aneg_t, 253

active, 253
block_lock, 255
complete, 253
fec_enable, 254
lp_aneg_able, 255
request_10g, 254
request_1g, 254
request_fec_change, 254
sm, 254
vtss_phy_10g_kr_status_fec_t, 255
corrected_block_cnt, 256
enable, 256
uncorrected_block_cnt, 256
vtss_phy_10g_kr_status_get
 vtss_phy_10g_api.h, 861
vtss_phy_10g_kr_status_train_t, 256
 c0_ob_tap_result, 257
 cm_ob_tap_result, 257
 complete, 257
 cp_ob_tap_result, 257
vtss_phy_10g_lane_sync_conf_t, 258
 enable, 258
 rx_ch, 259
 rx_macro, 258
 tx_ch, 259
 tx_macro, 258
vtss_phy_10g_lane_sync_set
 vtss_phy_10g_api.h, 856
vtss_phy_10g_line_clk_conf_set
 vtss_phy_10g_api.h, 855
vtss_phy_10g_line_clk_conf_t, 259
 clk_sel_no, 260
 mode, 260
 recvrd_clk_sel, 260
vtss_phy_10g_line_recvrd_clk_conf_set
 vtss_phy_10g_api.h, 856
vtss_phy_10g_loopback_get
 vtss_phy_10g_api.h, 865
vtss_phy_10g_loopback_set
 vtss_phy_10g_api.h, 864
vtss_phy_10g_loopback_t, 260
 enable, 261
 lb_type, 261
vtss_phy_10g_media_t
 vtss_phy_10g_api.h, 826
vtss_phy_10g_mode_get
 vtss_phy_10g_api.h, 842
vtss_phy_10g_mode_set
 vtss_phy_10g_api.h, 844
vtss_phy_10g_mode_t, 261
 apc_eqz_offs_par_cfg, 268
 apc_host_eqz_id_ctrl, 268
 apc_host_id_ctrl, 267
 apc_ib_regulator, 269
 apc_line_eqz_id_ctrl, 268
 apc_line_id_ctrl, 267
 apc_offs_ctrl, 266
 cfg0, 267
channel_id, 264
d_filter, 267
ddr_mode, 270
dig_offset_reg, 266
edc_fw_load, 265
enable_pass_thru, 273
h_apc_conf, 272
h_clk_src, 271
h_ib_conf, 272
h_media, 272
h_offset_guard, 269
high_input_gain, 264
hl_clk_synth, 264
ib_conf, 266
ib_ini_lp, 267
ib_max_lp, 268
ib_min_lp, 268
interface, 263
is_host_wan, 271
is_init, 273
l_apc_conf, 273
l_clk_src, 271
l_ib_conf, 272
l_media, 272
l_offset_guard, 269
link_6g_distance, 271
lref_for_host, 271
master, 270
ob_conf, 266
oper_mode, 263
pma_txratecontrol, 269
polarity, 270
rate, 270
rcvrd_clk, 265
rcvrd_clk_div, 265
sd6g_calib_done, 273
serdes_conf, 269
sref_clk_div, 265
use_conf, 266
venice_rev_a_los_detection_workaround, 270
wref_clk_div, 265
wrefclk, 263
xauı_lane_flip, 264
xfi_pol_invert, 264
vtss_phy_10g_ob_status_get
 vtss_phy_10g_api.h, 861
vtss_phy_10g_ob_status_t, 274
 c_ctrl, 274
 d_filt, 275
 is_host, 276
 levn, 275
 r_ctrl, 274
 slew, 274
 v3, 275
 v4, 275
 v5, 276
 vp, 275
vtss_phy_10g_pcs_prbs_gen_conf_get

vtss_phy_10g_api.h, 870
 vtss_phy_10g_pcs_prbs_gen_conf_set
 vtss_phy_10g_api.h, 870
 vtss_phy_10g_pcs_prbs_gen_conf_t, 276
 prbs_gen, 277
 vtss_phy_10g_pcs_prbs_mon_conf_get
 vtss_phy_10g_api.h, 871
 vtss_phy_10g_pcs_prbs_mon_conf_set
 vtss_phy_10g_api.h, 871
 vtss_phy_10g_pcs_prbs_mon_conf_t, 277
 error_counter, 277
 prbs_mon, 277
 vtss_phy_10g_pcs_prbs_mon_status_get
 vtss_phy_10g_api.h, 872
 vtss_phy_10g_pcs_status_get
 vtss_phy_10g_api.h, 880
 vtss_phy_10g_pkt_gen_conf
 vtss_phy_10g_api.h, 875
 vtss_phy_10g_pkt_gen_conf_t, 278
 dmac, 280
 enable, 278
 etype, 279
 frame_single, 279
 frames, 279
 ingress, 279
 ipg_len, 280
 pkt_len, 280
 ptp, 279
 ptp_ts_ns, 281
 ptp_ts_sec, 280
 smac, 280
 srate, 281
 vtss_phy_10g_pkt_mon_conf
 vtss_phy_10g_api.h, 875
 vtss_phy_10g_pkt_mon_conf_t, 281
 bad_crc, 282
 ber, 283
 enable, 282
 frag, 283
 good_crc, 282
 lfault, 283
 reset, 282
 update, 282
 vtss_phy_10g_pkt_mon_counters_get
 vtss_phy_10g_api.h, 876
 vtss_phy_10g_pkt_mon_rst_t
 vtss_phy_10g_api.h, 836
 vtss_phy_10g_polarity_inv_t, 283
 host_rx, 284
 host_tx, 284
 line_rx, 284
 line_tx, 284
 vtss_phy_10g_poll_1sec
 vtss_phy_10g_api.h, 881
 vtss_phy_10g_power_get
 vtss_phy_10g_api.h, 866
 vtss_phy_10g_power_set
 vtss_phy_10g_api.h, 866

vtss_phy_10g_power_t
 vtss_phy_10g_api.h, 835
 vtss_phy_10g_prbs_gen_conf
 vtss_phy_10g_api.h, 872
 vtss_phy_10g_prbs_gen_conf_get
 vtss_phy_10g_api.h, 873
 vtss_phy_10g_prbs_gen_conf_t, 285
 enable, 285
 line, 286
 prbsn_tx_io, 286
 prbsn_tx_iw, 286
 prbsn_tx_sel, 285
 vtss_phy_10g_prbs_mon_conf
 vtss_phy_10g_api.h, 873
 vtss_phy_10g_prbs_mon_conf_get
 vtss_phy_10g_api.h, 874
 vtss_phy_10g_prbs_mon_conf_t, 286
 active, 290
 bist_mode, 289
 des_interface_width, 288
 enable, 287
 error_states, 288
 error_status, 289
 incomplete, 290
 instable, 290
 line, 287
 main_status, 289
 max_bist_frames, 287
 no_of_errors, 288
 no_sync, 290
 PRBS_status, 289
 prbs_check_input_invert, 288
 prbsn_sel, 288
 stuck_at_01, 290
 stuck_at_par, 289
 vtss_phy_10g_prbs_mon_status_get
 vtss_phy_10g_api.h, 874
 vtss_phy_10g_reccrd_clk_sel_t
 vtss_phy_10g_api.h, 832
 vtss_phy_10g_reset
 vtss_phy_10g_api.h, 862
 vtss_phy_10g_rx_macro_t
 vtss_phy_10g_api.h, 833
 vtss_phy_10g_rxckout_conf_t, 291
 mode, 291
 squench_on_lopc, 292
 squench_on_pcs_fault, 291
 vtss_phy_10g_rxckout_get
 vtss_phy_10g_api.h, 851
 vtss_phy_10g_rxckout_set
 vtss_phy_10g_api.h, 851
 vtss_phy_10g_sckout_conf_set
 vtss_phy_10g_api.h, 854
 vtss_phy_10g_sckout_conf_t, 292
 enable, 293
 freq, 293
 mode, 292
 squench_inv, 293

src, 293
vtss_phy_10g_sckout_freq_t
 vtss_phy_10g_api.h, 832
vtss_phy_10g_serdes_status_get
 vtss_phy_10g_api.h, 862
vtss_phy_10g_serdes_status_t, 294
 h_pcs, 300
 h_pll5g_fsm_lock, 295
 h_pll5g_fsm_stat, 295
 h_pll5g_gain, 295
 h_pll5g_lock_status, 295
 h_pma, 299
 h_rx_rcpll_fsm_status, 297
 h_rx_rcpll_lock_status, 296
 h_rx_rcpll_range, 296
 h_rx_rcpll_vco_load, 297
 h_tx_rcpll_fsm_status, 298
 h_tx_rcpll_lock_status, 298
 h_tx_rcpll_range, 298
 h_tx_rcpll_vco_load, 298
 l_pcs, 300
 l_pll5g_fsm_lock, 296
 l_pll5g_fsm_stat, 296
 l_pll5g_gain, 296
 l_pll5g_lock_status, 295
 l_pma, 300
 l_rx_rcpll_fsm_status, 298
 l_rx_rcpll_lock_status, 297
 l_rx_rcpll_range, 297
 l_rx_rcpll_vco_load, 297
 l_tx_rcpll_fsm_status, 299
 l_tx_rcpll_lock_status, 299
 l_tx_rcpll_range, 299
 l_tx_rcpll_vco_load, 299
rcomp, 294
wis, 300
vtss_phy_10g_sgmii_mode_set
 vtss_phy_10g_api.h, 884
vtss_phy_10g_squelch_src_t
 vtss_phy_10g_api.h, 830
vtss_phy_10g_srefclk_conf_get
 vtss_phy_10g_api.h, 853
vtss_phy_10g_srefclk_conf_set
 vtss_phy_10g_api.h, 853
vtss_phy_10g_srefclk_freq_t
 vtss_phy_10g_api.h, 828
vtss_phy_10g_srefclk_mode_t, 301
 enable, 301
 freq, 301
vtss_phy_10g_status_get
 vtss_phy_10g_api.h, 862
vtss_phy_10g_status_t, 301
 block_lock, 304
 hpcs, 303
 hpcs_1g, 303
 hpma, 302
 lopc_stat, 304
 lpca_1g, 303
 lpca_1g_ts_fifo_sync
 pcs, 302
 pma, 302
 status, 303
 wis, 302
 xs, 303
 vtss_phy_10g_sync_clkout_get
 vtss_phy_10g_api.h, 849
 vtss_phy_10g_sync_clkout_set
 vtss_phy_10g_api.h, 850
 vtss_phy_10g_timestamp_val_t, 304
 timestamp, 305
 vtss_phy_10g_tx_macro_t
 vtss_phy_10g_api.h, 833
 vtss_phy_10g_txckout_conf_t, 305
 mode, 305
 vtss_phy_10g_txckout_get
 vtss_phy_10g_api.h, 852
 vtss_phy_10g_txckout_set
 vtss_phy_10g_api.h, 852
 vtss_phy_10g_vsphere_conf_get
 vtss_phy_10g_api.h, 869
 vtss_phy_10g_vsphere_conf_set
 vtss_phy_10g_api.h, 869
 vtss_phy_10g_vsphere_conf_t, 306
 enable, 306
 error_thres, 306
 line, 306
 scan_type, 306
 vtss_phy_10g_vsphere_scan_conf_t, 307
 ber, 309
 line, 307
 x_count, 308
 x_incr, 308
 x_start, 307
 y_count, 308
 y_incr, 308
 y_start, 308
 vtss_phy_10g_vsphere_scan_status_get
 vtss_phy_10g_api.h, 869
 vtss_phy_10g_vsphere_scan_status_t, 309
 amp_range, 310
 error_free_x, 310
 error_free_y, 310
 errors, 310
 scan_conf, 309
 vtss_phy_10g_vsphere_scan_t
 vtss_phy_10g_api.h, 836
 vtss_phy_10g_xfp_clkout_get
 vtss_phy_10g_api.h, 850
 vtss_phy_10g_xfp_clkout_set
 vtss_phy_10g_api.h, 851
 vtss_phy_1588_csr_reg_read
 vtss_phy_ts_api.h, 1026
 vtss_phy_1588_csr_reg_write
 vtss_phy_ts_api.h, 1026
 vtss_phy_1588_debug_reg_read
 vtss_phy_ts_api.h, 1030

vtss_phy_ts_api.h, 1030
 vtss_phy_6g_link_partner_distance_t
 vtss_phy_10g_api.h, 827
 vtss_phy_aneg_t, 311
 asymmetric_pause, 312
 speed_100m_fdx, 312
 speed_100m_hdx, 311
 speed_10m_fdx, 311
 speed_10m_hdx, 311
 speed_1g_fdx, 312
 speed_1g_hdx, 312
 symmetric_pause, 312
 tx_remote_fault, 313
 vtss_phy_api.h
 lb_type, 915
 MAX_CFG_BUF_SIZE, 896
 MAX_STAT_BUF_SIZE, 896
 MAX_WOL_MAC_ADDR_SIZE, 907
 MAX_WOL_PASSWD_SIZE, 907
 MIN_WOL_PASSWD_SIZE, 907
 rgmii_skew_delay_psec_t, 910
 VTSS_PHY_ACTIPHY_PWR, 897
 VTSS_PHY_LINK_AMS_EV, 901
 VTSS_PHY_LINK_AUTO_NEG_COMPLETE_EV,
 902
 VTSS_PHY_LINK_AUTO_NEG_ERROR_EV, 902
 VTSS_PHY_LINK_DOWN_PWR, 897
 VTSS_PHY_LINK_EXT_EEE_LINKFAIL_EV, 905
 VTSS_PHY_LINK_EXT_EEE_WAIT_RX_TQ_EV,
 905
 VTSS_PHY_LINK_EXT_EEE_WAIT_TS_EV, 905
 VTSS_PHY_LINK_EXT_EEE_WAKE_ERR_EV,
 904
 VTSS_PHY_LINK_EXT_MACSEC_EGRESS_EV,
 906
 VTSS_PHY_LINK_EXT_MACSEC_FC_BUFF_←
 EV, 906
 VTSS_PHY_LINK_EXT_MACSEC_HOST_MAC←
 EV, 905
 VTSS_PHY_LINK_EXT_MACSEC_INGRESS_←
 EV, 906
 VTSS_PHY_LINK_EXT_MACSEC_LINE_MAC_←
 EV, 906
 VTSS_PHY_LINK_EXT_MEM_INT_RING_EV,
 906
 VTSS_PHY_LINK_EXT_RR_SW_COMPL_EV,
 905
 VTSS_PHY_LINK_EXTENDED_REG_INT_EV,
 904
 VTSS_PHY_LINK_FALSE_CARRIER_INT_EV,
 903
 VTSS_PHY_LINK_FDX_STATE_CHANGE_EV,
 902
 VTSS_PHY_LINK_FFAIL_EV, 901
 VTSS_PHY_LINK_INLINE_POW_DEV_DETECT←
 T_EV, 902
 VTSS_PHY_LINK_LINK_SPEED_DS_DETECT←
 _EV, 903
 VTSS_PHY_LINK_LOS_EV, 901
 VTSS_PHY_LINK_MASTER_SLAVE_RES_ER←
 R_EV, 904
 VTSS_PHY_LINK_RX_ER_INT_EV, 904
 VTSS_PHY_LINK_RX_FIFO_OVERFLOW_INT←
 _EV, 903
 VTSS_PHY_LINK_SPEED_STATE_CHANGE←
 EV, 902
 VTSS_PHY_LINK_SYMBOL_ERR_INT_EV, 903
 VTSS_PHY_LINK_TX_FIFO_OVERFLOW_INT←
 _EV, 903
 VTSS_PHY_LINK_UP_FULL_PWR, 897
 VTSS_PHY_LINK_WAKE_ON_LAN_INT_EV, 904
 VTSS_PHY_PAGE_0x2DAF, 900
 VTSS_PHY_PAGE_1588, 899
 VTSS_PHY_PAGE_EXTENDED_2, 898
 VTSS_PHY_PAGE_EXTENDED_3, 898
 VTSS_PHY_PAGE_EXTENDED_4, 899
 VTSS_PHY_PAGE_EXTENDED, 898
 VTSS_PHY_PAGE_GPIO, 899
 VTSS_PHY_PAGE_MACSEC, 899
 VTSS_PHY_PAGE_STANDARD, 898
 VTSS_PHY_PAGE_TEST, 899
 VTSS_PHY_PAGE_TR, 900
 VTSS_PHY_POWER_ACTIPHY_BIT, 896
 VTSS_PHY_POWER_DYNAMIC_BIT, 896
 VTSS_PHY_RECov_CLK1, 897
 VTSS_PHY_RECov_CLK2, 897
 VTSS_PHY_RECov_CLK_NUM, 898
 VTSS_PHY_REG_EXTENDED, 900
 VTSS_PHY_REG_GPIO, 900
 VTSS_PHY_REG_STANDARD, 900
 VTSS_PHY_REG_TEST, 901
 VTSS_PHY_REG_TR, 901
 vga_adc_debug, 935
 vtss_fefi_mode_t, 915
 vtss_phy_atom12_patch_settings_get, 945
 vtss_phy_cfg_ib_cterm, 944
 vtss_phy_cfg_ob_post0, 944
 vtss_phy_chip_temp_get, 918
 vtss_phy_chip_temp_init, 918
 vtss_phy_cl37_lp_abil_get, 921
 vtss_phy_clk_source_t, 913
 vtss_phy_clk_squelch, 914
 vtss_phy_clock_conf_get, 925
 vtss_phy_clock_conf_set, 924
 vtss_phy_coma_mode_disable, 935
 vtss_phy_coma_mode_enable, 935
 vtss_phy_conf_1g_get, 922
 vtss_phy_conf_1g_set, 922
 vtss_phy_conf_get, 920
 vtss_phy_conf_set, 920
 vtss_phy_csr_rd, 941
 vtss_phy_csr_wr, 940
 vtss_phy_debug_phyinfo_print, 947
 vtss_phy_debug_regdump_print, 950
 vtss_phy_debug_register_dump, 948
 vtss_phy_debug_stat_print, 946

vtss_phy_detect_base_ports, 948
vtss_phy_do_page_chk_get, 942
vtss_phy_do_page_chk_set, 942
vtss_phy_eee_conf_get, 937
vtss_phy_eee_conf_set, 937
vtss_phy_eee_ena, 936
vtss_phy_eee_link_partner_advertisements_get, 938
vtss_phy_eee_power_save_state_get, 937
vtss_phy_enhanced_led_control_init, 934
vtss_phy_enhanced_led_control_init_get, 934
vtss_phy_event_enable_get, 939
vtss_phy_event_enable_set, 938
vtss_phy_event_poll, 939
vtss_phy_ext_connector_loopback, 949
vtss_phy_ext_event_poll, 960
vtss_phy_fast_link_fail_t, 911
vtss_phy_fefi_detect, 957
vtss_phy_fefi_get, 956
vtss_phy_fefi_set, 957
vtss_phy_flowcontrol_decode_status, 946
vtss_phy_forced_reset_t, 910
vtss_phy_freq_t, 914
vtss_phy_gpio_get, 931
vtss_phy_gpio_mode, 930
vtss_phy_gpio_mode_t, 914
vtss_phy_gpio_set, 931
vtss_phy_i2c_read, 925
vtss_phy_i2c_write, 926
vtss_phy_id_get, 921
vtss_phy_is_8051_crc_ok, 943
vtss_phy_is_viper_revB, 959
vtss_phy_lcpll_status_get, 953
vtss_phy_led_intensity, 907
vtss_phy_led_intensity_get, 933
vtss_phy_led_intensity_set, 933
vtss_phy_led_mode_set, 932
vtss_phy_led_mode_t, 908
vtss_phy_loopback_get, 943
vtss_phy_loopback_set, 943
vtss_phy_mac_media_inhibit_odd_start, 956
vtss_phy_mac_serdes_sgmmi_pre, 912
vtss_phy_macsec_csr_sd6g_rd, 961
vtss_phy_macsec_csr_sd6g_wr, 961
vtss_phy_mdi_t, 909
vtss_phy_media_force_ams_sel_t, 913
vtss_phy_media_interface_t, 909
vtss_phy_media_rem_fault_t, 912
vtss_phy_mmd_read, 928
vtss_phy_mmd_write, 928
vtss_phy_mode_t, 911
vtss_phy_mse_1000m_get, 958
vtss_phy_mse_100m_get, 958
vtss_phy_patch_settings_get, 952
vtss_phy_pkt_mode_t, 910
vtss_phy_port_eee_capable, 936
vtss_phy_post_reset, 916
vtss_phy_power_conf_get, 923
vtss_phy_power_conf_set, 924
vtss_phy_power_status_get, 924
vtss_phy_pre_reset, 916
vtss_phy_pre_system_reset, 917
vtss_phy_read, 927
vtss_phy_read_page, 927
vtss_phy_read_tr_addr, 959
vtss_phy_recov_clk_t, 907
vtss_phy_reg_decode_status, 945
vtss_phy_reset, 917
vtss_phy_reset_get, 918
vtss_phy_reset_lcpll, 954
vtss_phy_sd6g_ob_lev_rd, 955
vtss_phy_sd6g_ob_lev_wr, 956
vtss_phy_sd6g_ob_post_rd, 954
vtss_phy_sd6g_ob_post_wr, 955
vtss_phy_serdess1g_rcpll_status_get, 953
vtss_phy_serdess6g_rcpll_status_get, 952
vtss_phy_serdess_fmedia_loopback, 949
vtss_phy_serdess_sgmmi_loopback, 949
vtss_phy_sigdet_polarity_t, 911
vtss_phy_statistic_get, 941
vtss_phy_status_1g_get, 923
vtss_phy_status_get, 921
vtss_phy_status_inst_poll, 960
vtss_phy_unidirectional_t, 912
vtss_phy_veriphy_get, 932
vtss_phy_veriphy_start, 932
vtss_phy_warm_start_failed_get, 947
vtss_phy_wol_conf_get, 951
vtss_phy_wol_conf_set, 951
vtss_phy_wol_enable, 950
vtss_phy_write, 929
vtss_phy_write_masked, 929
vtss_phy_write_masked_page, 930
vtss_squelch_workaround, 940
vtss_wol_passwd_len_type_t, 915
vtss_phy_atom12_patch_settings_get
 vtss_phy_api.h, 945
vtss_phy_cfg_lb_cterm
 vtss_phy_api.h, 944
vtss_phy_cfg_ob_post0
 vtss_phy_api.h, 944
vtss_phy_chip_temp_get
 vtss_phy_api.h, 918
vtss_phy_chip_temp_init
 vtss_phy_api.h, 918
vtss_phy_cl37_lp_abil_get
 vtss_phy_api.h, 921
vtss_phy_clk_source_t
 vtss_phy_api.h, 913
vtss_phy_clk_squelch
 vtss_phy_api.h, 914
vtss_phy_clock_conf_get
 vtss_phy_api.h, 925
vtss_phy_clock_conf_set
 vtss_phy_api.h, 924
vtss_phy_clock_conf_t, 313

freq, 314
 squelch, 314
 src, 313
 vtss_phy_coma_mode_disable
 vtss_phy_api.h, 935
 vtss_phy_coma_mode_enable
 vtss_phy_api.h, 935
 vtss_phy_conf_1g_get
 vtss_phy_api.h, 922
 vtss_phy_conf_1g_set
 vtss_phy_api.h, 922
 vtss_phy_conf_1g_t, 314
 cfg, 315
 master, 315
 val, 315
 vtss_phy_conf_get
 vtss_phy_api.h, 920
 vtss_phy_conf_set
 vtss_phy_api.h, 920
 vtss_phy_conf_t, 315
 aneg, 316
 flf, 317
 force_ams_sel, 318
 forced, 316
 mac_if_pcs, 317
 mdi, 316
 media_if_pcs, 317
 mode, 316
 sigdet, 317
 unidir, 317
 vtss_phy_csr_rd
 vtss_phy_api.h, 941
 vtss_phy_csr_wr
 vtss_phy_api.h, 940
 vtss_phy_daisy_chain_conf_t, 318
 spi_daisy_input, 318
 spi_daisy_output, 319
 vtss_phy_daisy_conf_get
 vtss_phy_ts_api.h, 1008
 vtss_phy_daisy_conf_set
 vtss_phy_ts_api.h, 1007
 vtss_phy_debug_phyinfo_print
 vtss_phy_api.h, 947
 vtss_phy_debug_regdump_print
 vtss_phy_api.h, 950
 vtss_phy_debug_register_dump
 vtss_phy_api.h, 948
 vtss_phy_debug_stat_print
 vtss_phy_api.h, 946
 vtss_phy_detect_base_ports
 vtss_phy_api.h, 948
 vtss_phy_do_page_chk_get
 vtss_phy_api.h, 942
 vtss_phy_do_page_chk_set
 vtss_phy_api.h, 942
 vtss_phy_eee_conf_get
 vtss_phy_api.h, 937
 vtss_phy_eee_conf_set

 vtss_phy_api.h, 937
 vtss_phy_eee_conf_t, 319
 eee_ena_phy, 320
 eee_mode, 319
 vtss_phy_eee_ena
 vtss_phy_api.h, 936
 vtss_phy_eee_link_partner_advertisements_get
 vtss_phy_api.h, 938
 vtss_phy_eee_power_save_state_get
 vtss_phy_api.h, 937
 vtss_phy_enhanced_led_control_init
 vtss_phy_api.h, 934
 vtss_phy_enhanced_led_control_init_get
 vtss_phy_api.h, 934
 vtss_phy_enhanced_led_control_t, 320
 ser_led_frame_rate, 321
 ser_led_output_1, 320
 ser_led_output_2, 321
 ser_led_select, 321
 vtss_phy_event_enable_get
 vtss_phy_api.h, 939
 vtss_phy_event_enable_set
 vtss_phy_api.h, 938
 vtss_phy_event_poll
 vtss_phy_api.h, 939
 vtss_phy_ext_connector_loopback
 vtss_phy_api.h, 949
 vtss_phy_ext_event_poll
 vtss_phy_api.h, 960
 vtss_phy_fast_link_fail_t
 vtss_phy_api.h, 911
 vtss_phy_fefi_detect
 vtss_phy_api.h, 957
 vtss_phy_fefi_get
 vtss_phy_api.h, 956
 vtss_phy_fefi_set
 vtss_phy_api.h, 957
 vtss_phy_flowcontrol_decode_status
 vtss_phy_api.h, 946
 vtss_phy_forced_reset_t
 vtss_phy_api.h, 910
 vtss_phy_forced_t, 321
 fdx, 322
 speed, 322
 vtss_phy_freq_t
 vtss_phy_api.h, 914
 vtss_phy_gpio_get
 vtss_phy_api.h, 931
 vtss_phy_gpio_mode
 vtss_phy_api.h, 930
 vtss_phy_gpio_mode_t
 vtss_phy_api.h, 914
 vtss_phy_gpio_set
 vtss_phy_api.h, 931
 vtss_phy_i2c_read
 vtss_phy_api.h, 925
 vtss_phy_i2c_write
 vtss_phy_api.h, 926

vtss_phy_id_get
 vtss_phy_api.h, 921
vtss_phy_interface_mode
 vtss_phy_10g_api.h, 823
vtss_phy_is_8051_crc_ok
 vtss_phy_api.h, 943
vtss_phy_is_viper_revB
 vtss_phy_api.h, 959
vtss_phy_lcppll_status_get
 vtss_phy_api.h, 953
vtss_phy_led_intensity
 vtss_phy_api.h, 907
vtss_phy_led_intensity_get
 vtss_phy_api.h, 933
vtss_phy_led_intensity_set
 vtss_phy_api.h, 933
vtss_phy_led_mode_select_t, 322
 mode, 323
 number, 323
vtss_phy_led_mode_set
 vtss_phy_api.h, 932
vtss_phy_led_mode_t
 vtss_phy_api.h, 908
vtss_phy_loopback_get
 vtss_phy_api.h, 943
vtss_phy_loopback_set
 vtss_phy_api.h, 943
vtss_phy_loopback_t, 323
 connector_enable, 324
 far_end_enable, 324
 mac_serdes_equipment_enable, 325
 mac_serdes_facility_enable, 324
 mac_serdes_input_enable, 324
 media_serdes_equipment_enable, 325
 media_serdes_facility_enable, 325
 media_serdes_input_enable, 325
 near_end_enable, 324
vtss_phy_ltc_freq_synth_s, 326
 enable, 326
 high_duty_cycle, 326
 low_duty_cycle, 326
vtss_phy_mac_media_inhibit_odd_start
 vtss_phy_api.h, 956
vtss_phy_mac_serd_pcs_cntl_t, 327
 aneg_restart, 328
 disable, 327
 fast_link_stat_ena, 329
 force_adv_ability, 328
 inhibit_odd_start, 330
 pd_enable, 328
 restart, 328
 serdes_aneg_ena, 329
 serdes_pol_inv_in, 329
 serdes_pol_inv_out, 329
 sgmii_in_pre, 328
 sgmii_out_pre, 329
vtss_phy_mac_serd_pcs_sgmii_pre
 vtss_phy_api.h, 912
vtss_phy_macsec_csr_sd6g_rd
 vtss_phy_api.h, 961
vtss_phy_macsec_csr_sd6g_wr
 vtss_phy_api.h, 961
vtss_phy_mdi_t
 vtss_phy_api.h, 909
vtss_phy_media_force_ams_sel_t
 vtss_phy_api.h, 913
vtss_phy_media_interface_t
 vtss_phy_api.h, 909
vtss_phy_media_rem_fault_t
 vtss_phy_api.h, 912
vtss_phy_media_serd_pcs_cntl_t, 330
 aneg_pd_detect, 331
 force_adv_ability, 331
 force_fefi, 332
 force_fefi_value, 332
 force_hls, 332
 inhibit_odd_start, 331
 remote_fault, 331
 serdes_pol_inv_in, 331
 serdes_pol_inv_out, 331
vtss_phy_mmd_read
 vtss_phy_api.h, 928
vtss_phy_mmd_write
 vtss_phy_api.h, 928
vtss_phy_mode_t
 vtss_phy_api.h, 911
vtss_phy_mse_1000m_get
 vtss_phy_api.h, 958
vtss_phy_mse_100m_get
 vtss_phy_api.h, 958
vtss_phy_patch_settings_get
 vtss_phy_api.h, 952
vtss_phy_pcs_cnt_t, 332
 ber_cnt, 333
 block_lock_latched, 333
 err_blk_cnt, 333
 high_ber_latched, 333
vtss_phy_pkt_mode_t
 vtss_phy_api.h, 910
vtss_phy_port_eee_capable
 vtss_phy_api.h, 936
vtss_phy_post_reset
 vtss_phy_api.h, 916
vtss_phy_power_conf_get
 vtss_phy_api.h, 923
vtss_phy_power_conf_set
 vtss_phy_api.h, 924
vtss_phy_power_conf_t, 334
 mode, 334
vtss_phy_power_mode_t
 phy.h, 557
vtss_phy_power_status_get
 vtss_phy_api.h, 924
vtss_phy_power_status_t, 335
 level, 335
vtss_phy_pre_reset

vtss_phy_api.h, 916
 vtss_phy_pre_system_reset
 vtss_phy_api.h, 917
 vtss_phy_read
 vtss_phy_api.h, 927
 vtss_phy_read_page
 vtss_phy_api.h, 927
 vtss_phy_read_tr_addr
 vtss_phy_api.h, 959
 vtss_phy_recov_clk_t
 vtss_phy_api.h, 907
 vtss_phy_reg_decode_status
 vtss_phy_api.h, 945
 vtss_phy_reset
 vtss_phy_api.h, 917
 vtss_phy_reset_conf_t, 335
 force, 336
 i_cpu_en, 337
 mac_if, 336
 media_if, 336
 pkt_mode, 337
 rgmii, 336
 tbi, 336
 vtss_phy_reset_get
 vtss_phy_api.h, 918
 vtss_phy_reset_lcpll
 vtss_phy_api.h, 954
 vtss_phy_rgmii_conf_t, 337
 rx_clk_skew_ps, 338
 tx_clk_skew_ps, 338
 vtss_phy_sd6g_ob_lev_rd
 vtss_phy_api.h, 955
 vtss_phy_sd6g_ob_lev_wr
 vtss_phy_api.h, 956
 vtss_phy_sd6g_ob_post_rd
 vtss_phy_api.h, 954
 vtss_phy_sd6g_ob_post_wr
 vtss_phy_api.h, 955
 vtss_phy_serdes1g_rcpll_status_get
 vtss_phy_api.h, 953
 vtss_phy_serdes6g_rcpll_status_get
 vtss_phy_api.h, 952
 vtss_phy_serdes_fmedia_loopback
 vtss_phy_api.h, 949
 vtss_phy_serdes_sgmii_loopback
 vtss_phy_api.h, 949
 vtss_phy_sigdet_polarity_t
 vtss_phy_api.h, 911
 vtss_phy_statistic_get
 vtss_phy_api.h, 941
 vtss_phy_statistic_t, 338
 cu_bad, 339
 cu_good, 339
 media_mac_serdes_crc, 340
 media_mac_serdes_good, 340
 rx_err_cnt_base_tx, 339
 serdes_tx_bad, 339
 serdes_tx_good, 339
 vtss_phy_status_1g_get
 vtss_phy_api.h, 923
 vtss_phy_status_1g_t, 340
 master, 341
 master_cfg_fault, 341
 vtss_phy_status_get
 vtss_phy_api.h, 921
 vtss_phy_status_inst_poll
 vtss_phy_api.h, 960
 vtss_phy_tbi_conf_t, 341
 aneg_enable, 342
 vtss_phy_timestamp_t, 342
 high, 342
 low, 342
 nanoseconds, 343
 seconds, 343
 vtss_phy_ts_10g_extended_fifo_sync
 vtss_phy_ts_api.h, 1027
 vtss_phy_ts_10g_fifo_sync
 vtss_phy_ts_api.h, 1028
 vtss_phy_ts_8487_xaui_sel_t
 vtss_phy_ts_api.h, 985
 vtss_phy_ts_ach_conf_t, 343
 channel_type, 345
 comm_opt, 345
 mask, 344
 proto_id, 345
 value, 344
 version, 344
 vtss_phy_ts_action_format
 vtss_phy_ts_api.h, 990
 vtss_phy_ts_alt_clock_mode_get
 vtss_phy_ts_api.h, 994
 vtss_phy_ts_alt_clock_mode_s, 345
 ls_lpbk, 346
 ls_pps_lpbk, 346
 pps_ls_lpbk, 346
 vtss_phy_ts_alt_clock_mode_set
 vtss_phy_ts_api.h, 994
 vtss_phy_ts_alt_clock_mode_t
 vtss_phy_ts_api.h, 984
 vtss_phy_ts_alt_clock_saved_get
 vtss_phy_ts_api.h, 994
 vtss_phy_ts_api.h
 VTSS_PHY_TS_8487_XAUI_SEL_0, 982
 VTSS_PHY_TS_8487_XAUI_SEL_1, 982
 VTSS_PHY_TS_DATA_IN_RSRVD_FIELD, 981
 VTSS_PHY_TS_EGR_DATAPATH_RESET, 983
 VTSS_PHY_TS_EGR_ENGINE_ERR, 981
 VTSS_PHY_TS_EGR_FIFO_OVERFLOW, 981
 VTSS_PHY_TS_EGR_FIFO_RESET, 983
 VTSS_PHY_TS_EGR_LTC2_RESET, 983
 VTSS_PHY_TS_EGR_RW_FCS_ERR, 981
 VTSS_PHY_TS_EGR_TIMESTAMP_CAPTURED,
 981
 VTSS_PHY_TS_ENG_FLOW_VALID_FOR_CH0,
 979

VTSS_PHY_TS_ENG_FLOW_VALID_FOR_CH1,
980
VTSS_PHY_TS_ETH_ADDR_MATCH_48BIT, 975
VTSS_PHY_TS_ETH_ADDR_MATCH_ANY_M←
ULTICAST, 975
VTSS_PHY_TS_ETH_ADDR_MATCH_ANY_U←
NICAST, 975
VTSS_PHY_TS_ETH_MATCH_DEST_ADDR, 975
VTSS_PHY_TS_ETH_MATCH_SRC_ADDR, 975
VTSS_PHY_TS_ETH_MATCH_SRC_OR_DEST,
976
VTSS_PHY_TS_FIFO_SIG_DEST_IP, 971
VTSS_PHY_TS_FIFO_SIG_DEST_MAC, 972
VTSS_PHY_TS_FIFO_SIG_DOMAIN_NUM, 972
VTSS_PHY_TS_FIFO_SIG_MSG_TYPE, 971
VTSS_PHY_TS_FIFO_SIG_SEQ_ID, 972
VTSS_PHY_TS_FIFO_SIG_SOURCE_PORT_ID,
972
VTSS_PHY_TS_FIFO_SIG_SRC_IP, 971
VTSS_PHY_TS_INGR_DATAPATH_RESET, 982
VTSS_PHY_TS_INGR_ENGINE_ERR, 980
VTSS_PHY_TS_INGR_LTC1_RESET, 983
VTSS_PHY_TS_INGR_RW_FCS_ERR, 980
VTSS_PHY_TS_INGR_RW_PREAM_ERR, 980
VTSS_PHY_TS_IP_MATCH_DEST, 978
VTSS_PHY_TS_IP_MATCH_SRC_OR_DEST,
978
VTSS_PHY_TS_IP_MATCH_SRC, 978
VTSS_PHY_TS_IP_VER_4, 977
VTSS_PHY_TS_IP_VER_6, 977
VTSS_PHY_TS_LTC_LOAD_SAVE_NEW_TOD,
982
VTSS_PHY_TS_LTC_NEW_PPS_INTRPT, 982
VTSS_PHY_TS MPLS_STACK_DEPTH_1, 978
VTSS_PHY_TS MPLS_STACK_DEPTH_2, 978
VTSS_PHY_TS MPLS_STACK_DEPTH_3, 979
VTSS_PHY_TS MPLS_STACK_DEPTH_4, 979
VTSS_PHY_TS MPLS_STACK_REF_POINT←
END, 979
VTSS_PHY_TS MPLS_STACK_REF_POINT←
TOP, 979
VTSS_PHY_TS SIG_DEST_IP_LEN, 974
VTSS_PHY_TS SIG_DEST_MAC_LEN, 974
VTSS_PHY_TS SIG_DOMAIN_NUM_LEN, 973
VTSS_PHY_TS SIG_LEN, 972
VTSS_PHY_TS SIG_MSG_TYPE_LEN, 973
VTSS_PHY_TS SIG_SEQ_ID_LEN, 973
VTSS_PHY_TS SIG_SEQUENCE_ID_LEN, 974
VTSS_PHY_TS SIG_SOURCE_PORT_ID_LEN,
973
VTSS_PHY_TS SIG_SRC_IP_LEN, 974
VTSS_PHY_TS SIG_TIME_STAMP_LEN, 973
VTSS_PHY_TS TAG_RANGE_INNER, 977
VTSS_PHY_TS TAG_RANGE_NONE, 977
VTSS_PHY_TS TAG_RANGE_OUTER, 977
VTSS_PHY_TS TAG_TYPE_B, 976
VTSS_PHY_TS TAG_TYPE_C, 976
VTSS_PHY_TS TAG_TYPE_I, 976
VTSS_PHY_TS_TAG_TYPE_S, 976
VTSS_PTP_IP_1588_VERSION_2_1, 974
vtss_phy_1588_csr_reg_read, 1026
vtss_phy_1588_csr_reg_write, 1026
vtss_phy_1588_debug_reg_read, 1030
vtss_phy_1g_ts_fifo_sync, 1030
vtss_phy_daisy_conf_get, 1008
vtss_phy_daisy_conf_set, 1007
vtss_phy_ts_10g_extended_fifo_sync, 1027
vtss_phy_ts_10g_fifo_sync, 1028
vtss_phy_ts_8487_xaui_sel_t, 985
vtss_phy_ts_action_format, 990
vtss_phy_ts_alt_clock_mode_get, 994
vtss_phy_ts_alt_clock_mode_set, 994
vtss_phy_ts_alt_clock_mode_t, 984
vtss_phy_ts_alt_clock_saved_get, 994
vtss_phy_ts_block_soft_reset, 1024
vtss_phy_ts_bypass_clear, 1028
vtss_phy_ts_clock_rateadj_get, 1004
vtss_phy_ts_clock_rateadj_ppm_get, 1005
vtss_phy_ts_clock_rateadj_ppm_set, 1004
vtss_phy_ts_clock_rateadj_set, 1004
vtss_phy_ts_clock_src_t, 990
vtss_phy_ts_clockfreq_t, 990
vtss_phy_ts_correction_overflow_get, 1020
vtss_phy_ts_delay_asymmetry_get, 999
vtss_phy_ts_delay_asymmetry_set, 999
vtss_phy_ts_egress_engine_action_get, 1018
vtss_phy_ts_egress_engine_action_set, 1017
vtss_phy_ts_egress_engine_clear, 1014
vtss_phy_ts_egress_engine_conf_get, 1016
vtss_phy_ts_egress_engine_conf_set, 1015
vtss_phy_ts_egress_engine_init, 1012
vtss_phy_ts_egress_engine_init_conf_get, 1013
vtss_phy_ts_egress_latency_get, 997
vtss_phy_ts_egress_latency_set, 997
vtss_phy_ts_encap_t, 986
vtss_phy_ts_engine_channel_map_t, 984
vtss_phy_ts_engine_flow_match_t, 987
vtss_phy_ts_engine_t, 986
vtss_phy_ts_event_enable_get, 1019
vtss_phy_ts_event_enable_set, 1019
vtss_phy_ts_event_poll, 1019
vtss_phy_ts_event_t, 985
vtss_phy_ts_fifo_empty, 1009
vtss_phy_ts_fifo_mode_t, 992
vtss_phy_ts_fifo_read, 984
vtss_phy_ts_fifo_read_cb_get, 1010
vtss_phy_ts_fifo_read_install, 1010
vtss_phy_ts_fifo_sig_get, 1009
vtss_phy_ts_fifo_sig_mask_t, 984
vtss_phy_ts_fifo_sig_set, 1008
vtss_phy_ts_fifo_status_t, 986
vtss_phy_ts_fifo_timestamp_len_t, 992
vtss_phy_ts_flow_clear_cf_set, 1031
vtss_phy_ts_hiacc_get, 1024
vtss_phy_ts_hiacc_set, 1023

vtss_phy_ts_ieft_mpls_ach_oam_delaym_type_t, 989
 vtss_phy_ts_ieft_mpls_ach_oam_ts_format_t, 989
 vtss_phy_ts_ingress_engine_action_get, 1017
 vtss_phy_ts_ingress_engine_action_set, 1016
 vtss_phy_ts_ingress_engine_clear, 1012
 vtss_phy_ts_ingress_engine_conf_get, 1015
 vtss_phy_ts_ingress_engine_conf_set, 1014
 vtss_phy_ts_ingress_engine_init, 1011
 vtss_phy_ts_ingress_engine_init_conf_get, 1011
 vtss_phy_ts_ingress_latency_get, 996
 vtss_phy_ts_ingress_latency_set, 996
 vtss_phy_ts_init, 1022
 vtss_phy_ts_init_conf_get, 1022
 vtss_phy_ts_load_ptptime_get, 1002
 vtss_phy_ts_loadpulse_delay_get, 1003
 vtss_phy_ts_loadpulse_delay_set, 1003
 vtss_phy_ts_ltc_freq_synth_pulse_get, 1007
 vtss_phy_ts_ltc_freq_synth_pulse_set, 1007
 vtss_phy_ts_mode_get, 1021
 vtss_phy_ts_mode_set, 1021
 vtss_phy_ts_new_spi_mode_get, 1025
 vtss_phy_ts_new_spi_mode_set, 1024
 vtss_phy_ts_nphase_sampler_t, 993
 vtss_phy_ts_nphase_status_get, 1023
 vtss_phy_ts_ongoing_adjustment, 1006
 vtss_phy_ts_path_delay_get, 998
 vtss_phy_ts_path_delay_set, 998
 vtss_phy_ts_phy_oper_mode_change, 1025
 vtss_phy_ts_pps_conf_get, 995
 vtss_phy_ts_pps_conf_set, 995
 vtss_phy_ts_ptp_clock_mode_t, 988
 vtss_phy_ts_ptp_delaym_type_t, 988
 vtss_phy_ts_ptp_message_type_t, 993
 vtss_phy_ts_ptptime_adj1ns, 1005
 vtss_phy_ts_ptptime_arm, 1001
 vtss_phy_ts_ptptime_get, 1001
 vtss_phy_ts_ptptime_set, 1000
 vtss_phy_ts_ptptime_set_done, 1000
 vtss_phy_ts_rxtimestamp_len_t, 991
 vtss_phy_ts_rxtimestamp_pos_t, 991
 vtss_phy_ts_scaled_ppb_t, 984
 vtss_phy_ts_sertod_get, 1002
 vtss_phy_ts_sertod_set, 1002
 vtss_phy_ts_soft_reset_t, 985
 vtss_phy_ts_stats_get, 1020
 vtss_phy_ts_status_check, 1027
 vtss_phy_ts_tc_op_mode_t, 992
 vtss_phy_ts_tesla_tsp_fifo_sync, 1029
 vtss_phy_ts_timeofday_offset_set, 1006
 vtss_phy_ts_todadj_status_t, 985
 vtss_phy_ts_viper_fifo_reset, 1029
 vtss_phy_ts_y1731_oam_delaym_type_t, 988
 vtss_phy_ts_block_soft_reset
 vtss_phy_ts_api.h, 1024
 vtss_phy_ts_bypass_clear
 vtss_phy_ts_api.h, 1028
 vtss_phy_ts_clock_rateadj_get
 vtss_phy_ts_api.h, 1004
 vtss_phy_ts_clock_rateadj_ppm_get
 vtss_phy_ts_api.h, 1005
 vtss_phy_ts_clock_rateadj_ppm_set
 vtss_phy_ts_api.h, 1004
 vtss_phy_ts_clock_rateadj_set
 vtss_phy_ts_api.h, 1004
 vtss_phy_ts_clock_src_t
 vtss_phy_ts_api.h, 990
 vtss_phy_ts_clockfreq_t
 vtss_phy_ts_api.h, 990
 vtss_phy_ts_correction_overflow_get
 vtss_phy_ts_api.h, 1020
 vtss_phy_ts_delay_asymmetry_get
 vtss_phy_ts_api.h, 999
 vtss_phy_ts_delay_asymmetry_set
 vtss_phy_ts_api.h, 999
 vtss_phy_ts_egress_engine_action_get
 vtss_phy_ts_api.h, 1018
 vtss_phy_ts_egress_engine_action_set
 vtss_phy_ts_api.h, 1017
 vtss_phy_ts_egress_engine_clear
 vtss_phy_ts_api.h, 1014
 vtss_phy_ts_egress_engine_conf_get
 vtss_phy_ts_api.h, 1016
 vtss_phy_ts_egress_engine_conf_set
 vtss_phy_ts_api.h, 1015
 vtss_phy_ts_egress_engine_init
 vtss_phy_ts_api.h, 1012
 vtss_phy_ts_egress_engine_init_conf_get
 vtss_phy_ts_api.h, 1013
 vtss_phy_ts_egress_latency_get
 vtss_phy_ts_api.h, 997
 vtss_phy_ts_egress_latency_set
 vtss_phy_ts_api.h, 997
 vtss_phy_ts_encap_t
 vtss_phy_ts_api.h, 986
 vtss_phy_ts_eng_init_conf_t, 346
 encap_type, 347
 eng_used, 347
 flow_end_index, 348
 flow_match_mode, 347
 flow_st_index, 347
 vtss_phy_ts_engine_action_t, 348
 action, 349
 action_gen, 349
 action_ptp, 349
 gen_conf, 349
 oam_conf, 349
 ptp_conf, 349
 vtss_phy_ts_engine_channel_map_t
 vtss_phy_ts_api.h, 984
 vtss_phy_ts_engine_flow_conf_t, 350
 channel_map, 350
 eng_mode, 350
 flow_conf, 351
 gen, 351
 oam, 351

ptp, 351
vtss_phy_ts_engine_flow_match_t
 vtss_phy_ts_api.h, 987
vtss_phy_ts_engine_t
 vtss_phy_ts_api.h, 986
vtss_phy_ts_eth_conf_t, 352
 addr_match_mode, 354
 addr_match_select, 354
 comm_opt, 353
 etype, 353
 flow_en, 353
 flow_opt, 357
 i_tag, 357
 inner_tag, 357
 inner_tag_type, 355
 lower, 355
 mac_addr, 354
 mask, 356, 357
 num_tag, 354
 outer_tag, 356
 outer_tag_type, 355
 pbb_en, 353
 range, 356
 tag_range_mode, 355
 tpid, 353
 upper, 355
 val, 356, 357
 value, 356, 357
 vlan_check, 354
vtss_phy_ts_event_enable_get
 vtss_phy_ts_api.h, 1019
vtss_phy_ts_event_enable_set
 vtss_phy_ts_api.h, 1019
vtss_phy_ts_event_poll
 vtss_phy_ts_api.h, 1019
vtss_phy_ts_event_t
 vtss_phy_ts_api.h, 985
vtss_phy_ts_fifo_conf_t, 358
 detect_only, 358
 eng_minE, 359
 eng_recov, 358
 skip_rev_check, 359
vtss_phy_ts_fifo_empty
 vtss_phy_ts_api.h, 1009
vtss_phy_ts_fifo_mode_t
 vtss_phy_ts_api.h, 992
vtss_phy_ts_fifo_read
 vtss_phy_ts_api.h, 984
vtss_phy_ts_fifo_read_cb_get
 vtss_phy_ts_api.h, 1010
vtss_phy_ts_fifo_read_install
 vtss_phy_ts_api.h, 1010
vtss_phy_ts_fifo_sig_get
 vtss_phy_ts_api.h, 1009
vtss_phy_ts_fifo_sig_mask_t
 vtss_phy_ts_api.h, 984
vtss_phy_ts_fifo_sig_set
 vtss_phy_ts_api.h, 1008
vtss_phy_ts_fifo_sig_t, 359
 dest_ip, 361
 dest_mac, 361
 domain_num, 360
 msg_type, 360
 sequence_id, 360
 sig_mask, 360
 src_ip, 361
 src_port_identity, 360
vtss_phy_ts_fifo_status_t
 vtss_phy_ts_api.h, 986
vtss_phy_ts_fifo_timestamp_len_t
 vtss_phy_ts_api.h, 992
vtss_phy_ts_flow_clear_cf_set
 vtss_phy_ts_api.h, 1031
vtss_phy_ts_gen_conf_t, 361
 comm_opt, 362
 data, 363
 flow_en, 362
 flow_offset, 362
 flow_opt, 363
 mask, 363
 next_prot_offset, 362
vtss_phy_ts_generic_action_t, 363
 channel_map, 364
 data, 364
 enable, 364
 flow_id, 364
 mask, 365
 ts_offset, 365
 ts_type, 365
vtss_phy_ts_generic_flow_conf_t, 365
 eth1_opt, 366
 gen_opt, 366
vtss_phy_ts_hiacc_get
 vtss_phy_ts_api.h, 1024
vtss_phy_ts_hiacc_set
 vtss_phy_ts_api.h, 1023
vtss_phy_ts_ietf_mpls_ach_oam_conf_t, 366
 delaym_type, 367
 ds, 367
 ts_format, 367
vtss_phy_ts_ietf_mpls_ach_oam_delaym_type_t
 vtss_phy_ts_api.h, 989
vtss_phy_ts_ietf_mpls_ach_oam_ts_format_t
 vtss_phy_ts_api.h, 989
vtss_phy_ts_ingress_engine_action_get
 vtss_phy_ts_api.h, 1017
vtss_phy_ts_ingress_engine_action_set
 vtss_phy_ts_api.h, 1016
vtss_phy_ts_ingress_engine_clear
 vtss_phy_ts_api.h, 1012
vtss_phy_ts_ingress_engine_conf_get
 vtss_phy_ts_api.h, 1015
vtss_phy_ts_ingress_engine_conf_set
 vtss_phy_ts_api.h, 1014
vtss_phy_ts_ingress_engine_init
 vtss_phy_ts_api.h, 1011

vtss_phy_ts_ingress_engine_init_conf_get
 vtss_phy_ts_api.h, 1011
 vtss_phy_ts_ingress_latency_get
 vtss_phy_ts_api.h, 996
 vtss_phy_ts_ingress_latency_set
 vtss_phy_ts_api.h, 996
 vtss_phy_ts_init
 vtss_phy_ts_api.h, 1022
 vtss_phy_ts_init_conf_get
 vtss_phy_ts_api.h, 1022
 vtss_phy_ts_init_conf_t, 367
 auto_clear_ls, 370
 chk_ing_modified, 371
 clk_freq, 368
 clk_src, 368
 macsec_ena, 370
 one_step_txfifo, 371
 rx_ts_len, 369
 rx_ts_pos, 368
 tc_op_mode, 370
 tx_fifo_hi_clk_cycs, 369
 tx_fifo_lo_clk_cycs, 370
 tx_fifo_mode, 369
 tx_fifo_spi_conf, 369
 tx_ts_len, 369
 xaui_sel_8487, 370
 vtss_phy_ts_ip_conf_t, 371
 addr, 374
 comm_opt, 373
 dport_mask, 373
 dport_val, 373
 flow_en, 373
 flow_opt, 375
 ip_addr, 375
 ip_mode, 372
 ipv4, 374
 ipv6, 374
 mask, 374
 match_mode, 374
 sport_mask, 373
 sport_val, 372
 vtss_phy_ts_load_ptptime_get
 vtss_phy_ts_api.h, 1002
 vtss_phy_ts_loadpulse_delay_get
 vtss_phy_ts_api.h, 1003
 vtss_phy_ts_loadpulse_delay_set
 vtss_phy_ts_api.h, 1003
 vtss_phy_ts_ltc_freq_synth_pulse_get
 vtss_phy_ts_api.h, 1007
 vtss_phy_ts_ltc_freq_synth_pulse_set
 vtss_phy_ts_api.h, 1007
 vtss_phy_ts_mode_get
 vtss_phy_ts_api.h, 1021
 vtss_phy_ts_mode_set
 vtss_phy_ts_api.h, 1021
 vtss_phy_ts_mpls_conf_t, 375
 bottom_up, 378
 comm_opt, 376
 cw_en, 376
 end, 378
 flow_en, 376
 flow_opt, 379
 frst_lvl_after_top, 377
 frst_lvl_before_end, 378
 snd_lvl_after_top, 377
 snd_lvl_before_end, 378
 stack_depth, 376
 stack_level, 378
 stack_ref_point, 376
 thrd_lvl_after_top, 377
 thrd_lvl_before_end, 378
 top, 377
 top_down, 377
 vtss_phy_ts_mpls_lvl_rng_t, 379
 lower, 379
 upper, 379
 vtss_phy_ts_new_spi_mode_get
 vtss_phy_ts_api.h, 1025
 vtss_phy_ts_new_spi_mode_set
 vtss_phy_ts_api.h, 1024
 vtss_phy_ts_nphase_sampler_t
 vtss_phy_ts_api.h, 993
 vtss_phy_ts_nphase_status_get
 vtss_phy_ts_api.h, 1023
 vtss_phy_ts_nphase_status_t, 380
 CALIB_DONE, 381
 CALIB_ERR, 380
 enable, 380
 vtss_phy_ts_oam_engine_action_t, 381
 channel_map, 382
 enable, 382
 ietf_oam_conf, 382
 oam_conf, 383
 version, 382
 y1731_en, 382
 y1731_oam_conf, 382
 vtss_phy_ts_oam_engine_flow_conf_t, 383
 ach_opt, 384
 eth1_opt, 383
 eth2_opt, 384
 mpls_opt, 384
 vtss_phy_ts_ongoing_adjustment
 vtss_phy_ts_api.h, 1006
 vtss_phy_ts_path_delay_get
 vtss_phy_ts_api.h, 998
 vtss_phy_ts_path_delay_set
 vtss_phy_ts_api.h, 998
 vtss_phy_ts_phy_oper_mode_change
 vtss_phy_ts_api.h, 1025
 vtss_phy_ts_pps_conf_get
 vtss_phy_ts_api.h, 995
 vtss_phy_ts_pps_conf_set
 vtss_phy_ts_api.h, 995
 vtss_phy_ts_pps_config_s, 384
 pps_offset, 385
 pps_output_enable, 385

pps_width_adj, 385
vtss_phy_ts_ptp_clock_mode_t
 vtss_phy_ts_api.h, 988
vtss_phy_ts_ptp_conf_t, 386
 domain, 387
 lower, 387
 mask, 386
 range, 387
 range_en, 386
 upper, 387
 val, 386
 value, 387
vtss_phy_ts_ptp_delaym_type_t
 vtss_phy_ts_api.h, 988
vtss_phy_ts_ptp_engine_action_t, 388
 cf_update, 389
 channel_map, 388
 clk_mode, 389
 delaym_type, 389
 enable, 388
 ptp_conf, 388
vtss_phy_ts_ptp_engine_flow_conf_t, 389
 ach_opt, 391
 eth1_opt, 390
 eth2_opt, 390
 ip1_opt, 390
 ip2_opt, 390
 mpls_opt, 391
vtss_phy_ts_ptp_message_type_t
 vtss_phy_ts_api.h, 993
vtss_phy_ts_ptptime_adj1ns
 vtss_phy_ts_api.h, 1005
vtss_phy_ts_ptptime_arm
 vtss_phy_ts_api.h, 1001
vtss_phy_ts_ptptime_get
 vtss_phy_ts_api.h, 1001
vtss_phy_ts_ptptime_set
 vtss_phy_ts_api.h, 1000
vtss_phy_ts_ptptime_set_done
 vtss_phy_ts_api.h, 1000
vtss_phy_ts_rxtimestamp_len_t
 vtss_phy_ts_api.h, 991
vtss_phy_ts_rxtimestamp_pos_t
 vtss_phy_ts_api.h, 991
vtss_phy_ts_scaled_ppb_t
 vtss_phy_ts_api.h, 984
vtss_phy_ts_sertod_conf_t, 391
 ip_enable, 392
 ls_inv, 392
 op_enable, 392
vtss_phy_ts_sertod_get
 vtss_phy_ts_api.h, 1002
vtss_phy_ts_sertod_set
 vtss_phy_ts_api.h, 1002
vtss_phy_ts_soft_reset_t
 vtss_phy_ts_api.h, 985
vtss_phy_ts_stats_get
 vtss_phy_ts_api.h, 1020
vtss_phy_ts_stats_t, 392
 egr_fcs_err, 393
 egr_frm_mod_cnt, 394
 egr_pream_shrink_err, 393
 ingr_fcs_err, 393
 ingr_frm_mod_cnt, 394
 ingr_pream_shrink_err, 393
 ts_fifo_drop_cnt, 394
 ts_fifo_tx_cnt, 394
vtss_phy_ts_status_check
 vtss_phy_ts_api.h, 1027
vtss_phy_ts_tc_op_mode_t
 vtss_phy_ts_api.h, 992
vtss_phy_ts_tesla_tsp_fifo_sync
 vtss_phy_ts_api.h, 1029
vtss_phy_ts_timeofday_offset_set
 vtss_phy_ts_api.h, 1006
vtss_phy_ts_todadj_status_t
 vtss_phy_ts_api.h, 985
vtss_phy_ts_viper_fifo_reset
 vtss_phy_ts_api.h, 1029
vtss_phy_ts_y1731_oam_conf_t, 395
 delaym_type, 395
 lower, 396
 mask, 396
 meg_level, 396
 range, 396
 range_en, 395
 upper, 396
 val, 395
 value, 396
vtss_phy_ts_y1731_oam_delaym_type_t
 vtss_phy_ts_api.h, 988
vtss_phy_type_t, 397
 base_port_no, 398
 channel_id, 398
 part_number, 397
 phy_api_base_no, 398
 port_cnt, 398
 revision, 397
vtss_phy_unidirectional_t
 vtss_phy_api.h, 912
vtss_phy_veriphy_get
 vtss_phy_api.h, 932
vtss_phy_veriphy_result_t, 399
 length, 399
 link, 399
 status, 399
vtss_phy_veriphy_start
 vtss_phy_api.h, 932
vtss_phy_veriphy_status_t
 phy.h, 558
vtss_phy_warm_start_10g_failed_get
 vtss_phy_10g_api.h, 884
vtss_phy_warm_start_failed_get
 vtss_phy_api.h, 947
vtss_phy_wol_conf_get
 vtss_phy_api.h, 951

vtss_phy_wol_conf_set
 vtss_phy_api.h, 951
 vtss_phy_wol_conf_t, 400
 magic_pkt_cnt, 401
 secure_on_enable, 400
 wol_mac, 400
 wol_pass, 401
 wol_passwd_len, 401
 vtss_phy_wol_enable
 vtss_phy_api.h, 950
 vtss_phy_write
 vtss_phy_api.h, 929
 vtss_phy_write_masked
 vtss_phy_api.h, 929
 vtss_phy_write_masked_page
 vtss_phy_api.h, 930
 vtss_pi_conf_t, 401
 cs_wait_ns, 402
 vtss_policer_ext_t, 402
 broadcast, 403
 cpu_queue, 405
 dp_bypass_level, 403
 flooded, 404
 flow_control, 405
 frame_rate, 403
 learning, 404
 limit_cpu_traffic, 405
 limit_noncpu_traffic, 405
 mc_no_flood, 404
 multicast, 403
 to_cpu, 404
 uc_no_flood, 404
 unicast, 403
 vtss_policer_t, 406
 level, 406
 rate, 406
 vtss_policer_type_t
 types.h, 605
 vtss_poll_1sec
 vtss_misc_api.h, 732
 vtss_port_api.h
 CHIP_PORT_UNUSED, 1034
 VTSS_FRAME_GAP_DEFAULT, 1034
 VTSS_MAX_FRAME_LENGTH_MAX, 1035
 VTSS_MAX_FRAME_LENGTH_STANDARD,
 1034
 vtss_internal_bw_t, 1035
 vtss_miim_controller_t, 1035
 vtss_miim_read, 1044
 vtss_miim_write, 1044
 vtss_mmd_read, 1047
 vtss_mmd_write, 1047
 vtss_port_basic_counters_get, 1042
 vtss_port_clause_37_control_get, 1038
 vtss_port_clause_37_control_set, 1039
 vtss_port_clause_37_remote_fault_t, 1036
 vtss_port_conf_get, 1039
 vtss_port_conf_set, 1039
 vtss_port_counters_clear, 1041
 vtss_port_counters_get, 1041
 vtss_port_counters_update, 1040
 vtss_port_forward_state_get, 1042
 vtss_port_forward_state_set, 1042
 vtss_port_forward_t, 1037
 vtss_port_ifh_conf_get, 1043
 vtss_port_ifh_conf_set, 1043
 vtss_port_loop_t, 1037
 vtss_port_map_get, 1038
 vtss_port_map_set, 1037
 vtss_port_max_tags_t, 1036
 vtss_port_mmd_masked_write, 1046
 vtss_port_mmd_read, 1045
 vtss_port_mmd_read_inc, 1045
 vtss_port_mmd_write, 1046
 vtss_port_status_get, 1040
 vtss_port_basic_counters_get
 vtss_port_api.h, 1042
 vtss_port_bridge_counters_t, 406
 dot1dTpPortInDiscards, 407
 vtss_port_clause_37_adv_t, 407
 acknowledge, 408
 asymmetric_pause, 408
 fdx, 408
 hdx, 408
 next_page, 409
 remote_fault, 408
 symmetric_pause, 408
 vtss_port_clause_37_control_get
 vtss_port_api.h, 1038
 vtss_port_clause_37_control_set
 vtss_port_api.h, 1039
 vtss_port_clause_37_control_t, 409
 advertisement, 410
 enable, 409
 vtss_port_clause_37_remote_fault_t
 vtss_port_api.h, 1036
 vtss_port_conf_get
 vtss_port_api.h, 1039
 vtss_port_conf_set
 vtss_port_api.h, 1039
 vtss_port_conf_t, 410
 exc_col_cont, 413
 fdx, 412
 flow_control, 412
 frame_gaps, 411
 frame_length_chk, 413
 if_type, 411
 loop, 414
 max_frame_length, 412
 max_tags, 413
 power_down, 412
 sd_active_high, 411
 sd_enable, 411
 sd_internal, 411
 serdes, 414
 speed, 412

xaui_rx_lane_flip, 413
xaui_tx_lane_flip, 413
vtss_port_counters_clear
 vtss_port_api.h, 1041
vtss_port_counters_get
 vtss_port_api.h, 1041
vtss_port_counters_t, 414
 bridge, 415
 ethernet_like, 415
 if_group, 415
 prop, 415
 rmon, 415
vtss_port_counters_update
 vtss_port_api.h, 1040
vtss_port_ethernet_like_counters_t, 416
 dot3ControlInUnknownOpcodes, 417
 dot3InPauseFrames, 417
 dot3OutPauseFrames, 419
 dot3StatsAlignmentErrors, 416
 dot3StatsCarrierSenseErrors, 419
 dot3StatsDeferredTransmissions, 418
 dot3StatsExcessiveCollisions, 418
 dot3StatsFCSErrors, 417
 dot3StatsFrameTooLongs, 417
 dot3StatsLateCollisions, 418
 dot3StatsMultipleCollisionFrames, 418
 dot3StatsSingleCollisionFrames, 418
 dot3StatsSymbolErrors, 417
vtss_port_flow_control_conf_t, 419
 generate, 420
 obey, 420
 pfc, 420
 smac, 420
vtss_port_forward_state_get
 vtss_port_api.h, 1042
vtss_port_forward_state_set
 vtss_port_api.h, 1042
vtss_port_forward_t
 vtss_port_api.h, 1037
vtss_port_frame_gaps_t, 421
 fdx_gap, 421
 hdx_gap_1, 421
 hdx_gap_2, 421
vtss_port_if_group_counters_t, 422
 ifInBroadcastPkts, 423
 ifInDiscards, 423
 ifInErrors, 424
 ifInMulticastPkts, 423
 ifInNUcastPkts, 423
 ifInOctets, 422
 ifInUcastPkts, 423
 ifOutBroadcastPkts, 424
 ifOutDiscards, 425
 ifOutErrors, 425
 ifOutMulticastPkts, 424
 ifOutNUcastPkts, 425
 ifOutOctets, 424
 ifOutUcastPkts, 424
vtss_port_ifh_conf_get
 vtss_port_api.h, 1043
vtss_port_ifh_conf_set
 vtss_port_api.h, 1043
vtss_port_ifh_t, 425
 ena_ifh_header, 426
vtss_port_interface_t
 types.h, 603
vtss_port_loop_t
 vtss_port_api.h, 1037
vtss_port_map_get
 vtss_port_api.h, 1038
vtss_port_map_set
 vtss_port_api.h, 1037
vtss_port_map_t, 426
 chip_no, 427
 chip_port, 427
 miim_addr, 427
 miim_chip_no, 427
 miim_controller, 427
vtss_port_max_tags_t
 vtss_port_api.h, 1036
vtss_port_mmd_masked_write
 vtss_port_api.h, 1046
vtss_port_mmd_read
 vtss_port_api.h, 1045
vtss_port_mmd_read_inc
 vtss_port_api.h, 1045
vtss_port_mmd_write
 vtss_port_api.h, 1046
vtss_port_mux_mode_t
 vtss_init_api.h, 632
vtss_port_proprietary_counters_t, 428
 rx_prio, 428
 tx_prio, 428
vtss_port_rmon_counters_t, 429
 rx_etherStatsBroadcastPkts, 430
 rx_etherStatsCRCAlignErrors, 431
 rx_etherStatsDropEvents, 430
 rx_etherStatsFragments, 431
 rx_etherStatsJabbers, 431
 rx_etherStatsMulticastPkts, 430
 rx_etherStatsOctets, 430
 rx_etherStatsOversizePkts, 431
 rx_etherStatsPkts, 430
 rx_etherStatsPkts1024to1518Octets, 433
 rx_etherStatsPkts128to255Octets, 432
 rx_etherStatsPkts1519toMaxOctets, 433
 rx_etherStatsPkts256to511Octets, 432
 rx_etherStatsPkts512to1023Octets, 432
 rx_etherStatsPkts64Octets, 432
 rx_etherStatsPkts65to127Octets, 432
 rx_etherStatsUndersizePkts, 431
 tx_etherStatsBroadcastPkts, 434
 tx_etherStatsCollisions, 434
 tx_etherStatsDropEvents, 433
 tx_etherStatsMulticastPkts, 434
 tx_etherStatsOctets, 433

tx_etherStatsPkts, 433
 tx_etherStatsPkts1024to1518Octets, 435
 tx_etherStatsPkts128to255Octets, 435
 tx_etherStatsPkts1519toMaxOctets, 435
 tx_etherStatsPkts256to511Octets, 435
 tx_etherStatsPkts512to1023Octets, 435
 tx_etherStatsPkts64Octets, 434
 tx_etherStatsPkts65to127Octets, 434
 vtss_port_serdes_conf_t, 436
 sfp_dac, 436
 vtss_port_sgmii_aneg_t, 437
 aneg_complete, 438
 fdx, 437
 hdx, 437
 link, 437
 speed_100M, 438
 speed_10M, 438
 speed_1G, 438
 vtss_port_speed_t
 port.h, 570
 vtss_port_state_get
 vtss_l2_api.h, 665
 vtss_port_state_set
 vtss_l2_api.h, 666
 vtss_port_status_get
 vtss_port_api.h, 1040
 vtss_port_status_t, 439
 aneg, 440
 aneg_complete, 440
 copper, 441
 fdx, 440
 fiber, 441
 link, 439
 link_down, 439
 mdi_cross, 441
 remote_fault, 440
 speed, 439
 unidirectional_ability, 440
 vtss_ptp_event_enable
 vtss_misc_api.h, 732
 vtss_ptp_event_poll
 vtss_misc_api.h, 732
 vtss_pvlan_port_members_get
 vtss_l2_api.h, 680
 vtss_pvlan_port_members_set
 vtss_l2_api.h, 680
 vtss_qce_action_t, 441
 dp, 442
 dp_enable, 442
 dscp, 443
 dscp_enable, 443
 prio, 442
 prio_enable, 442
 vtss_qce_add
 vtss_qos_api.h, 1055
 vtss_qce_del
 vtss_qos_api.h, 1055
 vtss_qce_frame_etype_t, 443
 data, 444
 etype, 444
 vtss_qce_frame_ipv4_t, 444
 dport, 445
 dsop, 445
 fragment, 445
 proto, 445
 sip, 445
 sport, 445
 vtss_qce_frame_ipv6_t, 446
 dport, 447
 dscp, 446
 proto, 446
 sip, 447
 sport, 447
 vtss_qce_frame_llc_t, 447
 data, 448
 vtss_qce_frame_snap_t, 448
 data, 448
 vtss_qce_init
 vtss_qos_api.h, 1054
 vtss_qce_key_t, 449
 etype, 450
 frame, 451
 ipv4, 451
 ipv6, 451
 llc, 450
 mac, 449
 port_list, 449
 snap, 450
 tag, 450
 type, 450
 vtss_qce_mac_t, 451
 dmac_bc, 452
 dmac_mc, 452
 smac, 452
 vtss_qce_t, 453
 action, 453
 id, 453
 key, 453
 vtss_qce_tag_t, 454
 dei, 455
 pcp, 454
 tagged, 455
 vid, 454
 vtss_qce_type_t
 vtss_qos_api.h, 1052
 vtss_qos_api.h
 VTSS_PORT_POLICER_CPU_QUEUES, 1050
 VTSS_PORT_POLICERS, 1050
 VTSS_QCE_ID_LAST, 1051
 VTSS_QCL_ARRAY_SIZE, 1051
 VTSS_QCL_ID_END, 1051
 VTSS_QCL_ID_START, 1051
 VTSS_QCL_IDS, 1050
 vtss_dscp_emode_t, 1052
 vtss_dscp_mode_t, 1052
 vtss_qce_add, 1055

vtss_qce_del, 1055
vtss_qce_init, 1054
vtss_qce_type_t, 1052
vtss_qos_conf_get, 1053
vtss_qos_conf_set, 1053
vtss_qos_port_conf_get, 1054
vtss_qos_port_conf_set, 1054
vtss_tag_remark_mode_t, 1051
vtss_qos_conf_get
 vtss_qos_api.h, 1053
vtss_qos_conf_set
 vtss_qos_api.h, 1053
vtss_qos_conf_t, 455
 dscp_dp_level_map, 456
 dscp_qos_class_map, 456
 dscp_qos_map, 456
 dscp_remap, 457
 dscp_remark, 457
 dscp_translate_map, 457
 dscp_trust, 456
 prios, 456
vtss_qos_port_conf_get
 vtss_qos_api.h, 1054
vtss_qos_port_conf_set
 vtss_qos_api.h, 1054
vtss_qos_port_conf_t, 457
 default_dei, 460
 default_dpl, 460
 default_prio, 460
 dp_level_map, 461
 dscp_class_enable, 461
 dscp_emode, 461
 dscp_mode, 461
 dscp_translate, 462
 dwrr_enable, 463
 excess_enable, 459
 policer_ext_port, 459
 policer_port, 458
 policer_queue, 459
 qos_class_map, 461
 queue_pct, 463
 red, 458
 shaper_port, 459
 shaper_queue, 459
 tag_class_enable, 460
 tag_default_dei, 462
 tag_default_pcp, 462
 tag_dei_map, 463
 tag_pcp_map, 462
 tag_remark_mode, 462
 usr_prio, 460
vtss_qs_conf_get
 vtss_init_api.h, 637
vtss_qs_conf_set
 vtss_init_api.h, 636
vtss_qs_conf_t, 463
 mode, 464
 oversubscription, 464
 port, 465
 port_max, 464
 port_min, 464
 queue_max, 465
 queue_min, 465
vtss_qs_mode_t
 vtss_init_api.h, 632
vtss_rcpll_status_t, 465
 cal_error, 466
 cal_not_done, 466
 out_of_range, 466
vtss_recvrd_clkout_t
 vtss_phy_10g_api.h, 828
vtss_recvrd_t
 vtss_phy_10g_api.h, 824
vtss_recvrdclk_cdr_div_t
 vtss_phy_10g_api.h, 824
vtss_red_t, 467
 enable, 467
 max_prob_1, 468
 max_prob_2, 468
 max_prob_3, 468
 max_th, 467
 min_th, 467
vtss_reg_read
 vtss_misc_api.h, 729
vtss_reg_read_t
 vtss_init_api.h, 625
vtss_reg_write
 vtss_misc_api.h, 730
vtss_reg_write_masked
 vtss_misc_api.h, 730
vtss_reg_write_t
 vtss_init_api.h, 626
vtss_restart_conf_end
 vtss_init_api.h, 635
vtss_restart_conf_get
 vtss_init_api.h, 636
vtss_restart_conf_set
 vtss_init_api.h, 636
vtss_restart_status_get
 vtss_init_api.h, 635
vtss_restart_status_t, 468
 cur_version, 469
 prev_version, 469
 restart, 469
vtss_restart_t
 vtss_init_api.h, 633
vtss_routing_entry_t, 470
 ipv4_uc, 470
 ipv6_uc, 470
 route, 471
 type, 470
 vlan, 471
vtss_rptr_rate_t
 vtss_phy_10g_api.h, 826
vtss_rx_master_timestamp_get
 vtss_ts_api.h, 1086

vtss_rx_timestamp_get
 vtss_ts_api.h, 1085
 vtss_rx_timestamp_id_release
 vtss_ts_api.h, 1085
 vtss_secure_on_passwd_t, 471
 passwd, 472
 vtss_security_api.h
 VTSS_ACE_ID_LAST, 1059
 VTSS_PORT_NO_ANY, 1060
 vtss_ace_add, 1065
 vtss_ace_bit_t, 1061
 vtss_ace_counter_clear, 1066
 vtss_ace_counter_get, 1066
 vtss_ace_del, 1065
 vtss_ace_init, 1065
 vtss_ace_type_t, 1060
 vtss_acl_policer_conf_get, 1062
 vtss_acl_policer_conf_set, 1062
 vtss_acl_port_conf_get, 1063
 vtss_acl_port_conf_set, 1063
 vtss_acl_port_counter_clear, 1064
 vtss_acl_port_counter_get, 1064
 vtss_auth_port_state_get, 1061
 vtss_auth_port_state_set, 1062
 vtss_auth_state_t, 1060
 vtss_serdes_macro_conf_t, 472
 ib_cterm_ena, 473
 qsgmii, 473
 serdes1g_vdd, 472
 serdes6g_vdd, 472
 vtss_serdes_mode_t
 types.h, 604
 vtss_sflow_port_conf_get
 vtss_l2_api.h, 682
 vtss_sflow_port_conf_set
 vtss_l2_api.h, 683
 vtss_sflow_port_conf_t, 473
 sampling_rate, 474
 type, 474
 vtss_sflow_sampling_rate_convert
 vtss_l2_api.h, 683
 vtss_sflow_type_t
 l2_types.h, 537
 vtss_sgpi_bmode_t
 vtss_misc_api.h, 724
 vtss_sgpi_conf_get
 vtss_misc_api.h, 737
 vtss_sgpi_conf_set
 vtss_misc_api.h, 737
 vtss_sgpi_conf_t, 474
 bit_count, 475
 bmode, 475
 port_conf, 475
 vtss_sgpi_event_enable
 vtss_misc_api.h, 738
 vtss_sgpi_event_poll
 vtss_misc_api.h, 738
 vtss_sgpi_mode_t

 vtss_misc_api.h, 724
 vtss_sgpi_port_conf_t, 475
 enabled, 476
 int_pol_high, 476
 mode, 476
 vtss_sgpi_port_data_t, 477
 value, 477
 vtss_sgpi_read
 vtss_misc_api.h, 737
 vtss_shaper_t, 477
 level, 478
 rate, 478
 vtss_spi_32bit_read_write_t
 vtss_init_api.h, 627
 vtss_spi_64bit_read_write_t
 vtss_init_api.h, 628
 vtss_spi_read_write_t
 vtss_init_api.h, 627
 vtss_squelch_workaround
 vtss_phy_api.h, 940
 vtss_srefclk_div_t
 vtss_phy_10g_api.h, 824
 vtss_storm_policer_mode_t
 types.h, 605
 vtss_stp_port_state_get
 vtss_l2_api.h, 666
 vtss_stp_port_state_set
 vtss_l2_api.h, 666
 vtss_stp_state_t
 vtss_l2_api.h, 653
 vtss_sublayer_status_t, 478
 link_down, 479
 rx_fault, 479
 rx_link, 479
 tx_fault, 479
 vtss_sync_api.h
 VTSS_SYNCE_CLK_MAX, 1068
 VTSS_SYNCE_CLK_A, 1068
 VTSS_SYNCE_CLK_B, 1068
 vtss_sync_clock_in_get, 1070
 vtss_sync_clock_in_set, 1070
 vtss_sync_clock_out_get, 1070
 vtss_sync_clock_out_set, 1069
 vtss_sync_divider_t, 1069
 vtss_sync_clock_in_get
 vtss_sync_api.h, 1070
 vtss_sync_clock_in_set
 vtss_sync_api.h, 1070
 vtss_sync_clock_in_t, 480
 enable, 480
 port_no, 480
 squelsh, 480
 vtss_sync_clock_out_get
 vtss_sync_api.h, 1070
 vtss_sync_clock_out_set
 vtss_sync_api.h, 1069
 vtss_sync_clock_out_t, 481
 divider, 481

enable, 481
vtss_sync_edivider_t
 vtss_sync_api.h, 1069
vtss_tag_mark_mode_t
 vtss_qos_api.h, 1051
vtss_tag_type_t
 vtss_packet_api.h, 777
vtss_target_type_t
 vtss_init_api.h, 631
vtss_tci_t, 482
 cfi, 482
 tagprio, 483
 vid, 482
vtss_temp_sensor_get
 vtss_misc_api.h, 742
vtss_temp_sensor_init
 vtss_misc_api.h, 742
vtss_timeofday_t, 483
 sec, 483, 484
vtss_timestamp_age
 vtss_ts_api.h, 1087
vtss_timestamp_t, 484
 nanoseconds, 485
 sec_msb, 484
 seconds, 485
vtss_tod_get_ns_cnt
 vtss_misc_api.h, 741
vtss_tod_set_ns_cnt_cb
 vtss_misc_api.h, 741
vtss_trace_conf_get
 vtss_misc_api.h, 725
vtss_trace_conf_set
 vtss_misc_api.h, 726
vtss_trace_conf_t, 485
 level, 486
vtss_trace_group_t
 vtss_misc_api.h, 721
vtss_trace_layer_t
 vtss_misc_api.h, 720
vtss_trace_level_t
 vtss_misc_api.h, 721
vtss_ts_adjtimer_get
 vtss_ts_api.h, 1078
vtss_ts_adjtimer_one_sec
 vtss_ts_api.h, 1076
vtss_ts_adjtimer_set
 vtss_ts_api.h, 1077
vtss_ts_api.h
 vtss_rx_master_timestamp_get, 1086
 vtss_rx_timestamp_get, 1085
 vtss_rx_timestamp_id_release, 1085
 vtss_timestamp_age, 1087
 vtss_ts_adjtimer_get, 1078
 vtss_ts_adjtimer_one_sec, 1076
 vtss_ts_adjtimer_set, 1077
 vtss_ts_delay_asymmetry_get, 1083
 vtss_ts_delay_asymmetry_set, 1082
 vtss_ts_egress_latency_get, 1082
vtss_ts_egress_latency_set, 1081
vtss_ts_external_clock_mode_get, 1078
vtss_ts_external_clock_mode_set, 1079
vtss_ts_external_clock_saved_get, 1079
vtss_ts_freq_offset_get, 1078
vtss_ts_ingress_latency_get, 1080
vtss_ts_ingress_latency_set, 1080
vtss_ts_internal_mode_get, 1084
vtss_ts_internal_mode_set, 1084
vtss_ts_ongoing_adjustment, 1076
vtss_ts_operation_mode_get, 1083
vtss_ts_operation_mode_set, 1083
vtss_ts_p2p_delay_get, 1081
vtss_ts_p2p_delay_set, 1080
vtss_ts_status_change, 1087
vtss_ts_timeofday_get, 1076
vtss_ts_timeofday_next_pps_get, 1077
vtss_ts_timeofday_offset_set, 1075
vtss_ts_timeofday_set, 1074
vtss_ts_timeofday_set_delta, 1075
vtss_tx_timestamp_idx_alloc, 1086
vtss_tx_timestamp_update, 1085
vtss_ts_delay_asymmetry_get
 vtss_ts_api.h, 1083
vtss_ts_delay_asymmetry_set
 vtss_ts_api.h, 1082
vtss_ts_egress_latency_get
 vtss_ts_api.h, 1082
vtss_ts_egress_latency_set
 vtss_ts_api.h, 1081
vtss_ts_ext_clock_mode_t, 486
 enable, 486
 freq, 487
 one_pps_mode, 486
vtss_ts_external_clock_mode_get
 vtss_ts_api.h, 1078
vtss_ts_external_clock_mode_set
 vtss_ts_api.h, 1079
vtss_ts_external_clock_saved_get
 vtss_ts_api.h, 1079
vtss_ts_freq_offset_get
 vtss_ts_api.h, 1078
vtss_ts_id_t, 487
 ts_id, 487
vtss_ts_ingress_latency_get
 vtss_ts_api.h, 1080
vtss_ts_ingress_latency_set
 vtss_ts_api.h, 1080
vtss_ts_internal_mode_get
 vtss_ts_api.h, 1084
vtss_ts_internal_mode_set
 vtss_ts_api.h, 1084
vtss_ts_internal_mode_t, 488
 int_fmt, 488
vtss_ts_ongoing_adjustment
 vtss_ts_api.h, 1076
vtss_ts_operation_mode_get
 vtss_ts_api.h, 1083

vtss_ts_operation_mode_set
 vtss_ts_api.h, 1083
 vtss_ts_operation_mode_t, 489
 mode, 489
 vtss_ts_p2p_delay_get
 vtss_ts_api.h, 1081
 vtss_ts_p2p_delay_set
 vtss_ts_api.h, 1080
 vtss_ts_status_change
 vtss_ts_api.h, 1087
 vtss_ts_timeofday_get
 vtss_ts_api.h, 1076
 vtss_ts_timeofday_next_pps_get
 vtss_ts_api.h, 1077
 vtss_ts_timeofday_offset_set
 vtss_ts_api.h, 1075
 vtss_ts_timeofday_set
 vtss_ts_api.h, 1074
 vtss_ts_timeofday_set_delta
 vtss_ts_api.h, 1075
 vtss_ts_timestamp_alloc_t, 489
 cb, 490
 context, 490
 port_mask, 490
 vtss_ts_timestamp_t, 490
 context, 491
 id, 491
 ts, 491
 ts_valid, 491
 vtss_tx_timestamp_idx_alloc
 vtss_ts_api.h, 1086
 vtss_tx_timestamp_update
 vtss_ts_api.h, 1085
 vtss_uc_flood_members_get
 vtss_l2_api.h, 692
 vtss_uc_flood_members_set
 vtss_l2_api.h, 693
 vtss_vcap_bit_t
 types.h, 606
 vtss_vcap_ip_t, 492
 mask, 492
 value, 492
 vtss_vcap_key_type_t
 types.h, 607
 vtss_vcap_u128_t, 493
 mask, 493
 value, 493
 vtss_vcap_u16_t, 494
 mask, 494
 value, 494
 vtss_vcap_u24_t, 495
 mask, 495
 value, 495
 vtss_vcap_u32_t, 496
 mask, 496
 value, 496
 vtss_vcap_u40_t, 497
 mask, 497
 value, 497
 vtss_vcap_u48_t, 498
 mask, 498
 value, 498
 vtss_vcap_u8_t, 499
 mask, 499
 value, 499
 vtss_vcap_udp_tcp_t, 500
 high, 501
 in_range, 500
 low, 500
 vtss_vcap_vid_t, 501
 mask, 502
 value, 501
 vtss_vcap_vr_t, 502
 high, 503
 low, 503
 mask, 503
 r, 504
 type, 503
 v, 503
 value, 503
 vr, 504
 vtss_vcap_vr_type_t
 types.h, 607
 vtss_vce_action_t, 504
 policy_no, 505
 vid, 504
 vtss_vce_add
 vtss_l2_api.h, 675
 vtss_vce_del
 vtss_l2_api.h, 675
 vtss_vce_frame_etype_t, 505
 data, 506
 etype, 505
 vtss_vce_frame_ipv4_t, 506
 dport, 507
 dscp, 507
 fragment, 506
 options, 507
 proto, 507
 sip, 507
 vtss_vce_frame_ipv6_t, 508
 dport, 509
 dscp, 508
 proto, 508
 sip, 509
 vtss_vce_frame_llc_t, 509
 data, 510
 vtss_vce_frame_snap_t, 510
 data, 510
 vtss_vce_init
 vtss_l2_api.h, 675
 vtss_vce_key_t, 511
 etype, 512
 frame, 513
 ipv4, 512
 ipv6, 513

llc, 512
mac, 511
port_list, 511
snap, 512
tag, 511
type, 512
vtss_vce_mac_t, 513
 dmac_bc, 514
 dmac_mc, 514
 smac, 514
vtss_vce_t, 514
 action, 515
 id, 515
 key, 515
vtss_vce_tag_t, 515
 dei, 516
 pcp, 516
 s_tag, 517
 tagged, 516
 vid, 516
vtss_vce_type_t
 vtss_l2_api.h, 654
vtss_vcl_port_conf_get
 vtss_l2_api.h, 674
vtss_vcl_port_conf_set
 vtss_l2_api.h, 674
vtss_vcl_port_conf_t, 517
 dmac_dip, 517
vtss_vdd_t
 types.h, 606
vtss_vid_mac_t, 518
 mac, 518
 vid, 518
vtss_vlan_conf_get
 vtss_l2_api.h, 669
vtss_vlan_conf_set
 vtss_l2_api.h, 669
vtss_vlan_conf_t, 519
 s_etype, 519
vtss_vlan_frame_t
 types.h, 605
vtss_vlan_port_conf_get
 vtss_l2_api.h, 669
vtss_vlan_port_conf_set
 vtss_l2_api.h, 671
vtss_vlan_port_conf_t, 520
 frame_type, 521
 ingress_filter, 521
 port_type, 520
 pvid, 520
 untagged_vid, 520
vtss_vlan_port_members_get
 vtss_l2_api.h, 671
vtss_vlan_port_members_set
 vtss_l2_api.h, 672
vtss_vlan_port_type_t
 vtss_l2_api.h, 654
vtss_vlan_tag_t, 521
 dei, 522
 pcp, 522
 tpid, 522
 vid, 522
vtss_vlan_trans_group_add
 vtss_l2_api.h, 676
vtss_vlan_trans_group_del
 vtss_l2_api.h, 676
vtss_vlan_trans_group_get
 vtss_l2_api.h, 677
vtss_vlan_trans_group_to_port_get
 vtss_l2_api.h, 678
vtss_vlan_trans_group_to_port_set
 vtss_l2_api.h, 677
vtss_vlan_trans_grp2vlan_conf_t, 523
 group_id, 523
 trans_vid, 523
 vid, 523
vtss_vlan_trans_port2grp_conf_t, 524
 group_id, 524
 ports, 524
vtss_vlan_tx_tag_get
 vtss_l2_api.h, 673
vtss_vlan_tx_tag_set
 vtss_l2_api.h, 673
vtss_vlan_tx_tag_t
 vtss_l2_api.h, 654
vtss_vlan_vid_conf_get
 vtss_l2_api.h, 672
vtss_vlan_vid_conf_set
 vtss_l2_api.h, 672
vtss_vlan_vid_conf_t, 525
 fid, 526
 learning, 525
 mirror, 525
vtss_vstax_conf_get
 vtss_l2_api.h, 699
vtss_vstax_conf_set
 vtss_l2_api.h, 700
vtss_vstax_conf_t, 526
 cmef_disable, 527
 port_0, 527
 port_1, 527
 upsid_0, 526
 upsid_1, 527
vtss_vstax_fwd_mode_t
 vtss_packet_api.h, 776
vtss_vstax_glag_entry_t, 528
 upsid, 528
 upspn, 528
vtss_vstax_glag_get
 vtss_l2_api.h, 687
vtss_vstax_glag_set
 vtss_l2_api.h, 687
vtss_vstax_master_upsid_get
 vtss_l2_api.h, 701
vtss_vstax_master_upsid_set
 vtss_l2_api.h, 702

vtss_vstax_port_conf_get
 vtss_l2_api.h, 700
 vtss_vstax_port_conf_set
 vtss_l2_api.h, 701
 vtss_vstax_port_conf_t, 528
 mirror, 529
 ttl, 529
 vtss_vstax_route_entry_t, 529
 stack_port_a, 530
 stack_port_b, 530
 vtss_vstax_route_table_t, 530
 table, 531
 topology_type, 531
 vtss_vstax_rx_header_t, 531
 glag_no, 532
 isdx, 533
 port_no, 532
 sp, 532
 upsid, 532
 valid, 532
 vtss_vstax_topology_set
 vtss_l2_api.h, 702
 vtss_vstax_topology_type_t
 vtss_l2_api.h, 656
 vtss_vstax_tx_header_t, 533
 chip_port, 535
 dp, 535
 fwd_mode, 534
 glag_no, 535
 keep_ttl, 535
 port_no, 534
 prio, 534
 queue_no, 535
 tci, 534
 ttl, 534
 upsid, 534
 vtss_vt_id_t
 vtss_l2_api.h, 653
 vtss_wis_api.h
 VTSS_EWIS_AISL_EV, 1094
 VTSS_EWIS_AISP_EV, 1095
 VTSS_EWIS_B1_NZ_EV, 1098
 VTSS_EWIS_B1_THRESH_EV, 1099
 VTSS_EWIS_B2_NZ_EV, 1098
 VTSS_EWIS_B2_THRESH_EV, 1099
 VTSS_EWIS_B3_NZ_EV, 1098
 VTSS_EWIS_B3_THRESH_EV, 1099
 VTSS_EWIS_FAIS_EV, 1094
 VTSS_EWIS_FERDIP_EV, 1097
 VTSS_EWIS_FEUNEQP_EV, 1096
 VTSS_EWIS_FPLM_EV, 1093
 VTSS_EWIS_HIGH_BER_EV, 1097
 VTSS_EWIS_LCDP_EV, 1095
 VTSS_EWIS_LOF_EV, 1094
 VTSS_EWIS_LOPC_EV, 1096
 VTSS_EWIS_LOPP_EV, 1095
 VTSS_EWIS_LOS_EV, 1094
 VTSS_EWIS_MODULE_EV, 1095
 VTSS_EWIS_PCS_RECEIVE_FAULT_PEND, 1097
 VTSS_EWIS_PLMP_EV, 1095
 VTSS_EWIS_RDIL_EV, 1094
 VTSS_EWIS_REIL_EV, 1097
 VTSS_EWIS_REIL_NZ_EV, 1098
 VTSS_EWIS_REIL_THRESH_EV, 1099
 VTSS_EWIS_REIP_EV, 1097
 VTSS_EWIS_REIP_NZ_EV, 1098
 VTSS_EWIS_REIP_THRESH_EV, 1099
 VTSS_EWIS_RXLOL_EV, 1096
 VTSS_EWIS_SEF_EV, 1093
 VTSS_EWIS_TXLOL_EV, 1096
 VTSS_EWIS_UNEQP_EV, 1096
 vtss_ewis_cons_act_get, 1109
 vtss_ewis_cons_act_set, 1108
 vtss_ewis_counter_get, 1115
 vtss_ewis_counter_threshold_get, 1117
 vtss_ewis_counter_threshold_set, 1116
 vtss_ewis_defects_get, 1113
 vtss_ewis_event_enable, 1102
 vtss_ewis_event_force, 1103
 vtss_ewis_event_poll, 1102
 vtss_ewis_event_poll_without_mask, 1103
 vtss_ewis_event_t, 1100
 vtss_ewis_exp_sl_set, 1110
 vtss_ewis_force_conf_get, 1105
 vtss_ewis_force_conf_set, 1104
 vtss_ewis_mode_get, 1108
 vtss_ewis_mode_set, 1107
 vtss_ewis_mode_t, 1101
 vtss_ewis_path_acti_get, 1115
 vtss_ewis_path_txti_get, 1111
 vtss_ewis_path_txti_set, 1111
 vtss_ewis_perf_cntr_mode_t, 1101
 vtss_ewis_perf_get, 1116
 vtss_ewis_perf_mode_get, 1117
 vtss_ewis_perf_mode_set, 1117
 vtss_ewis_prbs31_err_inj_set, 1112
 vtss_ewis_prbs31_err_inj_t, 1102
 vtss_ewis_reset, 1108
 vtss_ewis_section_acti_get, 1114
 vtss_ewis_section_txti_get, 1110
 vtss_ewis_section_txti_set, 1109
 vtss_ewis_static_conf_get, 1104
 vtss_ewis_static_conf_t, 1100
 vtss_ewis_status_get, 1114
 vtss_ewis_test_counter_get, 1113
 vtss_ewis_test_mode_get, 1112
 vtss_ewis_test_mode_set, 1111
 vtss_ewis_test_pattern_s, 1101
 vtss_ewis_tti_mode_t, 1100
 vtss_ewis_tx_oh_get, 1106
 vtss_ewis_tx_oh_passthru_get, 1107
 vtss_ewis_tx_oh_passthru_set, 1106
 vtss_ewis_tx_oh_set, 1105
 vtss_wol_mac_addr_t, 536
 addr, 536

vtss_wol_passwd_len_type_t
 vtss_phy_api.h, 915

vtss_wref_clk_div_t
 vtss_phy_10g_api.h, 825

vtss_wrefclk_t
 vtss_phy_10g_api.h, 823

warm_start_enable
 vtss_init_conf_t, 144

wis
 vtss_phy_10g_serdess_status_t, 300
 vtss_phy_10g_status_t, 302

wol_mac
 vtss_phy_wol_conf_t, 400

wol_pass
 vtss_phy_wol_conf_t, 401

wol_passwd_len
 vtss_phy_wol_conf_t, 401

wref_clk_div
 vtss_phy_10g_mode_t, 265

wrefclk
 vtss_phy_10g_mode_t, 263

x_count
 vtss_phy_10g_vscope_scan_conf_t, 308

x_incr
 vtss_phy_10g_vscope_scan_conf_t, 308

x_start
 vtss_phy_10g_vscope_scan_conf_t, 307

xaui_lane_flip
 vtss_phy_10g_mode_t, 264

xaui_rx_lane_flip
 vtss_port_conf_t, 413

xaui_sel_8487
 vtss_phy_ts_init_conf_t, 370

xaui_tx_lane_flip
 vtss_port_conf_t, 413

xfi_pol_invert
 vtss_phy_10g_mode_t, 264

xs
 vtss_phy_10g_status_t, 303

xtr_qu
 vtss_packet_rx_meta_t, 194

xtr_qu_mask
 vtss_packet_rx_info_t, 187

y1731_en
 vtss_phy_ts_oam_engine_action_t, 382

y1731_oam_conf
 vtss_phy_ts_oam_engine_action_t, 382

y_count
 vtss_phy_10g_vscope_scan_conf_t, 308

y_incr
 vtss_phy_10g_vscope_scan_conf_t, 308

y_start
 vtss_phy_10g_vscope_scan_conf_t, 308