

# Vitesse API

Generated by Doxygen 1.8.14



# Contents

<b>1</b>	<b>Ethernet Virtual Connections</b>	<b>1</b>
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
<b>2</b>	<b>Layer 2</b>	<b>5</b>
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

<b>3</b>	<b>Layer 3</b>	<b>13</b>
<b>4</b>	<b>Security</b>	<b>15</b>
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
<b>5</b>	<b>Data Structure Index</b>	<b>17</b>
5.1	Data Structures . . . . .	17
<b>6</b>	<b>File Index</b>	<b>29</b>
6.1	File List . . . . .	29
<b>7</b>	<b>Data Structure Documentation</b>	<b>31</b>
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	power_mode . . . . .	34
7.2.2.10	exc_col_cont . . . . .	34
7.2.2.11	adv_dis . . . . .	35
7.2.2.12	max_tags . . . . .	35
7.2.2.13	oper_up . . . . .	35
7.2.2.14	frame_length_chk . . . . .	35
7.3	serdes_fields_t Struct Reference . . . . .	35
7.3.1	Detailed Description . . . . .	36
7.3.2	Field Documentation . . . . .	36
7.3.2.1	ob_post0 . . . . .	36
7.3.2.2	ob_sr . . . . .	36
7.4	tag_vtss_fdma_list Struct Reference . . . . .	36
7.4.1	Detailed Description . . . . .	37
7.4.2	Field Documentation . . . . .	37
7.4.2.1	frm_ptr . . . . .	37
7.4.2.2	act_len . . . . .	38
7.4.2.3	alloc_ptr . . . . .	38
7.4.2.4	rx_info . . . . .	38
7.4.2.5	sw_tstamp . . . . .	38
7.4.2.6	user . . . . .	39
7.4.2.7	afi_frm_cnt . . . . .	39
7.4.2.8	afi_seq_number . . . . .	39
7.4.2.9	next . . . . .	39
7.5	vtss_ace_frame_arp_t Struct Reference . . . . .	40
7.5.1	Detailed Description . . . . .	40
7.5.2	Field Documentation . . . . .	40
7.5.2.1	smac . . . . .	40
7.5.2.2	arp . . . . .	40
7.5.2.3	req . . . . .	41

7.5.2.4	unknown	41
7.5.2.5	smac_match	41
7.5.2.6	dmac_match	41
7.5.2.7	length	41
7.5.2.8	ip	42
7.5.2.9	ethernet	42
7.5.2.10	sip	42
7.5.2.11	dip	42
7.6	vtss_ace_frame_etype_t Struct Reference	42
7.6.1	Detailed Description	43
7.6.2	Field Documentation	43
7.6.2.1	dmac	43
7.6.2.2	smac	43
7.6.2.3	etype	43
7.6.2.4	data	44
7.7	vtss_ace_frame_ipv4_t Struct Reference	44
7.7.1	Detailed Description	44
7.7.2	Field Documentation	44
7.7.2.1	ttl	45
7.7.2.2	fragment	45
7.7.2.3	options	45
7.7.2.4	ds	45
7.7.2.5	proto	45
7.7.2.6	sip	46
7.7.2.7	dip	46
7.7.2.8	data	46
7.7.2.9	sport	46
7.7.2.10	dport	46
7.7.2.11	tcp_fin	47
7.7.2.12	tcp_syn	47

7.7.2.13	tcp_rst	47
7.7.2.14	tcp_psh	47
7.7.2.15	tcp_ack	47
7.7.2.16	tcp_urg	48
7.7.2.17	sip_eq_dip	48
7.7.2.18	sport_eq_dport	48
7.7.2.19	seq_zero	48
7.8	vtss_ace_frame_ipv6_t Struct Reference	48
7.8.1	Detailed Description	49
7.8.2	Field Documentation	49
7.8.2.1	proto	49
7.8.2.2	sip	49
7.8.2.3	ttl	50
7.8.2.4	ds	50
7.8.2.5	data	50
7.8.2.6	sport	50
7.8.2.7	dport	50
7.8.2.8	tcp_fin	51
7.8.2.9	tcp_syn	51
7.8.2.10	tcp_RST	51
7.8.2.11	tcp_psh	51
7.8.2.12	tcp_ack	51
7.8.2.13	tcp_urg	52
7.8.2.14	sip_eq_dip	52
7.8.2.15	sport_eq_dport	52
7.8.2.16	seq_zero	52
7.9	vtss_ace_frame_llc_t Struct Reference	52
7.9.1	Detailed Description	53
7.9.2	Field Documentation	53
7.9.2.1	dmac	53

---

7.9.2.2	smac . . . . .	53
7.9.2.3	llc . . . . .	53
7.10	vtss_ace_frame_snap_t Struct Reference . . . . .	54
7.10.1	Detailed Description . . . . .	54
7.10.2	Field Documentation . . . . .	54
7.10.2.1	dmac . . . . .	54
7.10.2.2	smac . . . . .	54
7.10.2.3	snap . . . . .	55
7.11	vtss_ace_t Struct Reference . . . . .	55
7.11.1	Detailed Description . . . . .	55
7.11.2	Field Documentation . . . . .	55
7.11.2.1	id . . . . .	56
7.11.2.2	port_no . . . . .	56
7.11.2.3	policy . . . . .	56
7.11.2.4	type . . . . .	56
7.11.2.5	action . . . . .	56
7.11.2.6	dmac_mc . . . . .	57
7.11.2.7	dmac_bc . . . . .	57
7.11.2.8	vlan . . . . .	57
7.11.2.9	etype . . . . .	57
7.11.2.10	llc . . . . .	57
7.11.2.11	snap . . . . .	58
7.11.2.12	arp . . . . .	58
7.11.2.13	ipv4 . . . . .	58
7.11.2.14	ipv6 . . . . .	58
7.11.2.15	frame . . . . .	58
7.12	vtss_ace_vlan_t Struct Reference . . . . .	59
7.12.1	Detailed Description . . . . .	59
7.12.2	Field Documentation . . . . .	59
7.12.2.1	vid . . . . .	59

7.12.2.2 <code>usr_prio</code>	59
7.12.2.3 <code>cfi</code>	60
7.13 <code>vtss_acl_action_t</code> Struct Reference	60
7.13.1 Detailed Description	60
7.13.2 Field Documentation	60
7.13.2.1 <code>cpu</code>	60
7.13.2.2 <code>cpu_once</code>	61
7.13.2.3 <code>cpu_queue</code>	61
7.13.2.4 <code>police</code>	61
7.13.2.5 <code>policer_no</code>	61
7.13.2.6 <code>learn</code>	61
7.13.2.7 <code>forward</code>	62
7.13.2.8 <code>port_forward</code>	62
7.13.2.9 <code>port_no</code>	62
7.13.2.10 <code>irq_trigger</code>	62
7.14 <code>vtss_acl_policer_conf_t</code> Struct Reference	62
7.14.1 Detailed Description	63
7.14.2 Field Documentation	63
7.14.2.1 <code>rate</code>	63
7.15 <code>vtss_acl_port_conf_t</code> Struct Reference	63
7.15.1 Detailed Description	63
7.15.2 Field Documentation	64
7.15.2.1 <code>policy_no</code>	64
7.15.2.2 <code>action</code>	64
7.16 <code>vtss_aggr_mode_t</code> Struct Reference	64
7.16.1 Detailed Description	64
7.16.2 Field Documentation	65
7.16.2.1 <code>smac_enable</code>	65
7.16.2.2 <code>dmac_enable</code>	65
7.16.2.3 <code>sip_dip_enable</code>	65

7.16.2.4 <code>sport_dport_enable</code>	65
7.17 <code>vtss_aneg_t</code> Struct Reference	66
7.17.1 Detailed Description	66
7.17.2 Field Documentation	66
7.17.2.1 <code>obey_pause</code>	66
7.17.2.2 <code>generate_pause</code>	66
7.18 <code>vtss_api_lock_t</code> Struct Reference	66
7.18.1 Detailed Description	67
7.18.2 Field Documentation	67
7.18.2.1 <code>inst</code>	67
7.18.2.2 <code>function</code>	67
7.18.2.3 <code>file</code>	67
7.18.2.4 <code>line</code>	68
7.19 <code>vtss_basic_counters_t</code> Struct Reference	68
7.19.1 Detailed Description	68
7.19.2 Field Documentation	68
7.19.2.1 <code>rx_frames</code>	68
7.19.2.2 <code>tx_frames</code>	69
7.20 <code>vtss_chip_id_t</code> Struct Reference	69
7.20.1 Detailed Description	69
7.20.2 Field Documentation	69
7.20.2.1 <code>part_number</code>	69
7.20.2.2 <code>revision</code>	70
7.21 <code>vtss_counter_pair_t</code> Struct Reference	70
7.21.1 Detailed Description	70
7.21.2 Field Documentation	70
7.21.2.1 <code>frames</code>	70
7.21.2.2 <code>bytes</code>	71
7.22 <code>vtss_debug_info_t</code> Struct Reference	71
7.22.1 Detailed Description	71

---

7.22.2 Field Documentation . . . . .	71
7.22.2.1 layer . . . . .	71
7.22.2.2 group . . . . .	72
7.22.2.3 chip_no . . . . .	72
7.22.2.4 port_list . . . . .	72
7.22.2.5 full . . . . .	72
7.22.2.6 clear . . . . .	72
7.22.2.7 vml_format . . . . .	73
7.23 vtss_debug_lock_t Struct Reference . . . . .	73
7.23.1 Detailed Description . . . . .	73
7.23.2 Field Documentation . . . . .	73
7.23.2.1 chip_no . . . . .	73
7.24 vtss_dgroup_port_conf_t Struct Reference . . . . .	74
7.24.1 Detailed Description . . . . .	74
7.24.2 Field Documentation . . . . .	74
7.24.2.1 dgroup_no . . . . .	74
7.25 vtss_dlb_policer_conf_t Struct Reference . . . . .	74
7.25.1 Detailed Description . . . . .	75
7.25.2 Field Documentation . . . . .	75
7.25.2.1 type . . . . .	75
7.25.2.2 enable . . . . .	75
7.25.2.3 cm . . . . .	75
7.25.2.4 cf . . . . .	75
7.25.2.5 line_rate . . . . .	76
7.25.2.6 cir . . . . .	76
7.25.2.7 cbs . . . . .	76
7.25.2.8 eir . . . . .	76
7.25.2.9 ebs . . . . .	76
7.26 vtss_ece_action_t Struct Reference . . . . .	77
7.26.1 Detailed Description . . . . .	77

---

7.26.2 Field Documentation . . . . .	77
7.26.2.1 dir . . . . .	77
7.26.2.2 pop_tag . . . . .	77
7.26.2.3 outer_tag . . . . .	78
7.26.2.4 inner_tag . . . . .	78
7.26.2.5 policer_id . . . . .	78
7.26.2.6 evc_id . . . . .	78
7.26.2.7 policy_no . . . . .	78
7.27 vtss_ece_frame_ipv4_t Struct Reference . . . . .	79
7.27.1 Detailed Description . . . . .	79
7.27.2 Field Documentation . . . . .	79
7.27.2.1 dscp . . . . .	79
7.28 vtss_ece_frame_ipv6_t Struct Reference . . . . .	79
7.28.1 Detailed Description . . . . .	80
7.28.2 Field Documentation . . . . .	80
7.28.2.1 dscp . . . . .	80
7.29 vtss_ece_inner_tag_t Struct Reference . . . . .	80
7.29.1 Detailed Description . . . . .	80
7.29.2 Field Documentation . . . . .	80
7.29.2.1 type . . . . .	81
7.29.2.2 vid . . . . .	81
7.29.2.3 pcp_dei_preserve . . . . .	81
7.29.2.4 pcp . . . . .	81
7.29.2.5 dei . . . . .	81
7.30 vtss_ece_key_t Struct Reference . . . . .	82
7.30.1 Detailed Description . . . . .	82
7.30.2 Field Documentation . . . . .	82
7.30.2.1 port_list . . . . .	82
7.30.2.2 mac . . . . .	82
7.30.2.3 tag . . . . .	83

---

7.30.2.4  inner_tag . . . . .	83
7.30.2.5  type . . . . .	83
7.30.2.6  ipv4 . . . . .	83
7.30.2.7  ipv6 . . . . .	83
7.30.2.8  frame . . . . .	84
7.31  vtss_ece_mac_t Struct Reference . . . . .	84
7.31.1  Detailed Description . . . . .	84
7.31.2  Field Documentation . . . . .	84
7.31.2.1  dmac . . . . .	84
7.32  vtss_ece_outer_tag_t Struct Reference . . . . .	85
7.32.1  Detailed Description . . . . .	85
7.32.2  Field Documentation . . . . .	85
7.32.2.1  enable . . . . .	85
7.32.2.2  vid . . . . .	85
7.32.2.3  pcp_dei_preserve . . . . .	86
7.32.2.4  pcp . . . . .	86
7.32.2.5  dei . . . . .	86
7.33  vtss_ece_t Struct Reference . . . . .	86
7.33.1  Detailed Description . . . . .	86
7.33.2  Field Documentation . . . . .	87
7.33.2.1  id . . . . .	87
7.33.2.2  key . . . . .	87
7.33.2.3  action . . . . .	87
7.34  vtss_ece_tag_t Struct Reference . . . . .	87
7.34.1  Detailed Description . . . . .	88
7.34.2  Field Documentation . . . . .	88
7.34.2.1  vid . . . . .	88
7.34.2.2  pcp . . . . .	88
7.34.2.3  dei . . . . .	88
7.34.2.4  tagged . . . . .	88

---

7.34.2.5 <code>s_tagged</code>	89
7.35 <code>vtss_eee_port_conf_t</code> Struct Reference	89
7.35.1 Detailed Description	89
7.35.2 Field Documentation	89
7.35.2.1 <code>eee_ena</code>	89
7.35.2.2 <code>eee_fast_queues</code>	90
7.35.2.3 <code>tx_tw</code>	90
7.35.2.4 <code>lp_advertisement</code>	90
7.35.2.5 <code>optimized_for_power</code>	90
7.36 <code>vtss_eee_port_counter_t</code> Struct Reference	90
7.36.1 Detailed Description	91
7.36.2 Field Documentation	91
7.36.2.1 <code>fill_level_get</code>	91
7.36.2.2 <code>fill_level_thres</code>	91
7.36.2.3 <code>fill_level</code>	91
7.36.2.4 <code>tx_out_bytes_get</code>	92
7.36.2.5 <code>tx_out_bytes</code>	92
7.37 <code>vtss_eee_port_state_t</code> Struct Reference	92
7.37.1 Detailed Description	92
7.37.2 Field Documentation	92
7.37.2.1 <code>select</code>	93
7.37.2.2 <code>val</code>	93
7.38 <code>vtss_eps_port_conf_t</code> Struct Reference	93
7.38.1 Detailed Description	93
7.38.2 Field Documentation	93
7.38.2.1 <code>type</code>	94
7.38.2.2 <code>port_no</code>	94
7.39 <code>vtss_evc_conf_t</code> Struct Reference	94
7.39.1 Detailed Description	94
7.39.2 Field Documentation	94

---

---

7.39.2.1	policer_id	95
7.39.2.2	learning	95
7.39.2.3	pb	95
7.39.2.4	network	95
7.40	vtss_evc_counters_t Struct Reference	95
7.40.1	Detailed Description	96
7.40.2	Field Documentation	96
7.40.2.1	rx_green	96
7.40.2.2	rx_yellow	96
7.40.2.3	rx_red	96
7.40.2.4	rx_discard	97
7.40.2.5	tx_discard	97
7.40.2.6	tx_green	97
7.40.2.7	tx_yellow	97
7.41	vtss_evc_pb_conf_t Struct Reference	97
7.41.1	Detailed Description	98
7.41.2	Field Documentation	98
7.41.2.1	nni	98
7.41.2.2	ivid	98
7.41.2.3	vid	98
7.41.2.4	leaf_ivid	99
7.41.2.5	leaf_vid	99
7.41.2.6	leaf	99
7.42	vtss_evc_port_conf_t Struct Reference	99
7.42.1	Detailed Description	99
7.42.2	Field Documentation	100
7.42.2.1	dei_colouring	100
7.43	vtss_ewis_aisl_cons_act_s Struct Reference	100
7.43.1	Detailed Description	100
7.43.2	Field Documentation	100

---

7.43.2.1 ais_on_los . . . . .	100
7.43.2.2 ais_on_lof . . . . .	101
7.44 vtss_ewis_conf_s Struct Reference . . . . .	101
7.44.1 Detailed Description . . . . .	101
7.44.2 Field Documentation . . . . .	101
7.44.2.1 ewis_init_done . . . . .	102
7.44.2.2 static_conf . . . . .	102
7.44.2.3 ewis_mode . . . . .	102
7.44.2.4 section_cons_act . . . . .	102
7.44.2.5 section_txти . . . . .	102
7.44.2.6 force_mode . . . . .	103
7.44.2.7 path_txти . . . . .	103
7.44.2.8 tx_oh . . . . .	103
7.44.2.9 tx_oh_passthru . . . . .	103
7.44.2.10 exp_sl . . . . .	103
7.44.2.11 test_conf . . . . .	104
7.44.2.12 ewis_cntr_thresh_conf . . . . .	104
7.44.2.13 perf_mode . . . . .	104
7.45 vtss_ewis_cons_act_s Struct Reference . . . . .	104
7.45.1 Detailed Description . . . . .	104
7.45.2 Field Documentation . . . . .	105
7.45.2.1 aisl . . . . .	105
7.45.2.2 rdil . . . . .	105
7.45.2.3 fault . . . . .	105
7.46 vtss_ewis_counter_s Struct Reference . . . . .	105
7.46.1 Detailed Description . . . . .	106
7.46.2 Field Documentation . . . . .	106
7.46.2.1 pn_ebc_p . . . . .	106
7.46.2.2 pf_ebc_p . . . . .	106
7.46.2.3 pn_ebc_l . . . . .	106

---

7.46.2.4	pf_ebc_l	106
7.46.2.5	pn_ebc_s	107
7.47	vtss_ewis_counter_threshold_s Struct Reference	107
7.47.1	Detailed Description	107
7.47.2	Field Documentation	107
7.47.2.1	n_ebc_thr_s	107
7.47.2.2	n_ebc_thr_l	108
7.47.2.3	f_ebc_thr_l	108
7.47.2.4	n_ebc_thr_p	108
7.47.2.5	f_ebc_thr_p	108
7.48	vtss_ewis_defects_s Struct Reference	108
7.48.1	Detailed Description	109
7.48.2	Field Documentation	109
7.48.2.1	dlos_s	109
7.48.2.2	doof_s	109
7.48.2.3	dlaf_s	110
7.48.2.4	dais_l	110
7.48.2.5	drdi_l	110
7.48.2.6	dais_p	110
7.48.2.7	dlop_p	110
7.48.2.8	duneq_p	111
7.48.2.9	drdi_p	111
7.48.2.10	dlcd_p	111
7.48.2.11	dplm_p	111
7.48.2.12	dfais_p	111
7.48.2.13	dfplm_p	112
7.48.2.14	dfuneq_p	112
7.49	vtss_ewis_fault_cons_act_s Struct Reference	112
7.49.1	Detailed Description	112
7.49.2	Field Documentation	113

---

7.49.2.1 fault_on_feplmp . . . . .	113
7.49.2.2 fault_on_feaisp . . . . .	113
7.49.2.3 fault_on_rdiil . . . . .	113
7.49.2.4 fault_on_sef . . . . .	113
7.49.2.5 fault_on_lof . . . . .	113
7.49.2.6 fault_on_los . . . . .	114
7.49.2.7 fault_on_aisl . . . . .	114
7.49.2.8 fault_on_lcdp . . . . .	114
7.49.2.9 fault_on_plmp . . . . .	114
7.49.2.10 fault_on_aisp . . . . .	114
7.49.2.11 fault_on_lopp . . . . .	115
7.50 vtss_ewis_force_mode_s Struct Reference . . . . .	115
7.50.1 Detailed Description . . . . .	115
7.50.2 Field Documentation . . . . .	115
7.50.2.1 line_rx_force . . . . .	115
7.50.2.2 line_tx_force . . . . .	116
7.50.2.3 path_force . . . . .	116
7.51 vtss_ewis_line_force_mode_s Struct Reference . . . . .	116
7.51.1 Detailed Description . . . . .	116
7.51.2 Field Documentation . . . . .	116
7.51.2.1 force_ais . . . . .	117
7.51.2.2 force_rdi . . . . .	117
7.52 vtss_ewis_line_tx_force_mode_s Struct Reference . . . . .	117
7.52.1 Detailed Description . . . . .	117
7.52.2 Field Documentation . . . . .	117
7.52.2.1 force_ais . . . . .	118
7.52.2.2 force_rdi . . . . .	118
7.53 vtss_ewis_path_force_mode_s Struct Reference . . . . .	118
7.53.1 Detailed Description . . . . .	118
7.53.2 Field Documentation . . . . .	118

---

7.53.2.1 force_uneq . . . . .	119
7.53.2.2 force_rdi . . . . .	119
7.54 vtss_ewis_perf_mode_s Struct Reference . . . . .	119
7.54.1 Detailed Description . . . . .	119
7.54.2 Field Documentation . . . . .	119
7.54.2.1 pn_ebc_mode_s . . . . .	120
7.54.2.2 pn_ebc_mode_l . . . . .	120
7.54.2.3 pf_ebc_mode_l . . . . .	120
7.54.2.4 pn_ebc_mode_p . . . . .	120
7.54.2.5 pf_ebc_mode_p . . . . .	120
7.55 vtss_ewis_perf_s Struct Reference . . . . .	121
7.55.1 Detailed Description . . . . .	121
7.55.2 Field Documentation . . . . .	121
7.55.2.1 pn_ebc_s . . . . .	121
7.55.2.2 pn_ebc_l . . . . .	121
7.55.2.3 pf_ebc_l . . . . .	122
7.55.2.4 pn_ebc_p . . . . .	122
7.55.2.5 pf_ebc_p . . . . .	122
7.56 vtss_ewis_rdil_cons_act_s Struct Reference . . . . .	122
7.56.1 Detailed Description . . . . .	123
7.56.2 Field Documentation . . . . .	123
7.56.2.1 rdil_on_los . . . . .	123
7.56.2.2 rdil_on_lof . . . . .	123
7.56.2.3 rdil_on_lopc . . . . .	123
7.56.2.4 rdil_on_ais_l . . . . .	123
7.57 vtss_ewis_sl_conf_s Struct Reference . . . . .	124
7.57.1 Detailed Description . . . . .	124
7.57.2 Field Documentation . . . . .	124
7.57.2.1 exsl . . . . .	124
7.58 vtss_ewis_static_conf_s Struct Reference . . . . .	124

---

7.58.1	Detailed Description	125
7.58.2	Field Documentation	125
7.58.2.1	ewis_txctrl1	125
7.58.2.2	ewis_txctrl2	125
7.58.2.3	ewis_rx_ctrl1	126
7.58.2.4	ewis_mode_ctrl	126
7.58.2.5	ewis_tx_a1_a2	126
7.58.2.6	ewis_tx_c2_h1	126
7.58.2.7	ewis_tx_h2_h3	126
7.58.2.8	ewis_tx_z0_e1	127
7.58.2.9	ewis_rx_frm_ctrl1	127
7.58.2.10	ewis_rx_frm_ctrl2	127
7.58.2.11	ewis_lof_ctrl1	127
7.58.2.12	ewis_lof_ctrl2	127
7.58.2.13	ewis_rx_err_frc1	128
7.58.2.14	ewis_pmtick_ctrl	128
7.58.2.15	ewis_cnt_cfg	128
7.59	vtss_ewis_status_s Struct Reference	128
7.59.1	Detailed Description	128
7.59.2	Field Documentation	129
7.59.2.1	fault	129
7.59.2.2	link_stat	129
7.60	vtss_ewis_test_conf_s Struct Reference	129
7.60.1	Detailed Description	129
7.60.2	Field Documentation	130
7.60.2.1	loopback	130
7.60.2.2	test_pattern_gen	130
7.60.2.3	test_pattern_ana	130
7.61	vtss_ewis_test_status_s Struct Reference	130
7.61.1	Detailed Description	131

7.61.2 Field Documentation . . . . .	131
7.61.2.1 <code>tstpat_cnt</code> . . . . .	131
7.61.2.2 <code>ana_sync</code> . . . . .	131
7.62 <code>vtss_ewis_tti_s</code> Struct Reference . . . . .	131
7.62.1 Detailed Description . . . . .	132
7.62.2 Field Documentation . . . . .	132
7.62.2.1 <code>mode</code> . . . . .	132
7.62.2.2 <code>tti</code> . . . . .	132
7.62.2.3 <code>valid</code> . . . . .	132
7.63 <code>vtss_ewis_tx_oh_s</code> Struct Reference . . . . .	132
7.63.1 Detailed Description . . . . .	133
7.63.2 Field Documentation . . . . .	133
7.63.2.1 <code>tx_dcc_s</code> . . . . .	133
7.63.2.2 <code>tx_e1</code> . . . . .	133
7.63.2.3 <code>tx_f1</code> . . . . .	134
7.63.2.4 <code>tx_z0</code> . . . . .	134
7.63.2.5 <code>tx_dcc_l</code> . . . . .	134
7.63.2.6 <code>tx_e2</code> . . . . .	134
7.63.2.7 <code>tx_k1_k2</code> . . . . .	134
7.63.2.8 <code>tx_s1</code> . . . . .	135
7.63.2.9 <code>tx_z1_z2</code> . . . . .	135
7.63.2.10 <code>tx_c2</code> . . . . .	135
7.63.2.11 <code>tx_f2</code> . . . . .	135
7.63.2.12 <code>tx_n1</code> . . . . .	135
7.63.2.13 <code>tx_z3_z4</code> . . . . .	136
7.64 <code>vtss_ewis_tx_passthru_s</code> Struct Reference . . . . .	136
7.64.1 Detailed Description . . . . .	136
7.64.2 Field Documentation . . . . .	136
7.64.2.1 <code>tx_j0</code> . . . . .	137
7.64.2.2 <code>tx_z0</code> . . . . .	137

7.64.2.3 tx_b1 . . . . .	137
7.64.2.4 tx_e1 . . . . .	137
7.64.2.5 tx_f1 . . . . .	137
7.64.2.6 tx_dcc_s . . . . .	138
7.64.2.7 tx_soh . . . . .	138
7.64.2.8 tx_b2 . . . . .	138
7.64.2.9 tx_k1 . . . . .	138
7.64.2.10 tx_k2 . . . . .	138
7.64.2.11 tx_reil . . . . .	139
7.64.2.12 tx_dcc_l . . . . .	139
7.64.2.13 tx_s1 . . . . .	139
7.64.2.14 tx_e2 . . . . .	139
7.64.2.15 tx_z1_z2 . . . . .	139
7.64.2.16 tx_loh . . . . .	140
7.65 vtss_fan_conf_t Struct Reference . . . . .	140
7.65.1 Detailed Description . . . . .	140
7.65.2 Field Documentation . . . . .	140
7.65.2.1 fan_pwm_freq . . . . .	140
7.65.2.2 fan_low_pol . . . . .	141
7.65.2.3 fan_open_col . . . . .	141
7.65.2.4 type . . . . .	141
7.65.2.5 ppr . . . . .	141
7.66 vtss_fdma_cfg_t Struct Reference . . . . .	141
7.66.1 Detailed Description . . . . .	142
7.66.2 Field Documentation . . . . .	142
7.66.2.1 enable . . . . .	142
7.66.2.2 rx_mtu . . . . .	143
7.66.2.3 rx_buf_cnt . . . . .	143
7.66.2.4 rx_alloc_cb . . . . .	143
7.66.2.5 rx_dont_strip_vlan_tag . . . . .	144

---

7.66.2.6 rx_dont_reinsert_vlan_tag . . . . .	144
7.66.2.7 rx_allow_vlan_tag_mismatch . . . . .	145
7.66.2.8 rx_allow_multiple_dcbs . . . . .	145
7.66.2.9 rx_cb . . . . .	145
7.66.2.10 tx_buf_cnt . . . . .	146
7.66.2.11 tx_done_cb . . . . .	146
7.66.2.12 afi_buf_cnt . . . . .	146
7.66.2.13 afi_done_cb . . . . .	147
7.67 vtss_fdma_ch_cfg_t Struct Reference . . . . .	147
7.67.1 Detailed Description . . . . .	147
7.67.2 Field Documentation . . . . .	148
7.67.2.1 usage . . . . .	148
7.67.2.2 xtr_grp . . . . .	148
7.67.2.3 inj_grp_mask . . . . .	148
7.67.2.4 list . . . . .	149
7.67.2.5 xtr_cb . . . . .	149
7.67.2.6 prio . . . . .	150
7.67.2.7 chip_no . . . . .	150
7.67.2.8 ccm_quotient_max . . . . .	150
7.68 vtss_fdma_throttle_cfg_t Struct Reference . . . . .	151
7.68.1 Detailed Description . . . . .	151
7.68.2 Field Documentation . . . . .	151
7.68.2.1 frm_limit_per_tick . . . . .	151
7.68.2.2 byte_limit_per_tick . . . . .	152
7.68.2.3 suspend_tick_cnt . . . . .	152
7.69 vtss_fdma_tx_info_t Struct Reference . . . . .	152
7.69.1 Detailed Description . . . . .	152
7.69.2 Field Documentation . . . . .	153
7.69.2.1 pre_cb_ctxt1 . . . . .	153
7.69.2.2 pre_cb_ctxt2 . . . . .	153

7.69.2.3 pre_cb . . . . .	153
7.69.2.4 afi_fps . . . . .	154
7.69.2.5 afi_type . . . . .	154
7.69.2.6 afi_enable_counting . . . . .	154
7.69.2.7 afi_enable_sequence_numbering . . . . .	155
7.69.2.8 afi_sequence_number_offset . . . . .	155
7.70 vtss_gpio_10g_gpio_mode_t Struct Reference . . . . .	156
7.70.1 Detailed Description . . . . .	156
7.70.2 Field Documentation . . . . .	156
7.70.2.1 mode . . . . .	156
7.70.2.2 port . . . . .	156
7.70.2.3 input . . . . .	157
7.70.2.4 in_sig . . . . .	157
7.70.2.5 p_gpio . . . . .	157
7.70.2.6 c_intrpt . . . . .	157
7.70.2.7 source . . . . .	157
7.70.2.8 aggr_intrpt . . . . .	158
7.70.2.9 use_as_intrpt . . . . .	158
7.70.2.10 p_gpio_intrpt . . . . .	158
7.70.2.11 led_conf . . . . .	158
7.70.2.12 invert_output . . . . .	158
7.71 vtss_gpio_10g_led_conf_t Struct Reference . . . . .	159
7.71.1 Detailed Description . . . . .	159
7.71.2 Field Documentation . . . . .	159
7.71.2.1 blink . . . . .	159
7.72 vtss_init_conf_t Struct Reference . . . . .	159
7.72.1 Detailed Description . . . . .	160
7.72.2 Field Documentation . . . . .	160
7.72.2.1 reg_read . . . . .	160
7.72.2.2 reg_write . . . . .	160

---

7.72.2.3	miim_read	161
7.72.2.4	miim_write	161
7.72.2.5	mmd_read	161
7.72.2.6	mmd_read_inc	161
7.72.2.7	mmd_write	161
7.72.2.8	spi_read_write	162
7.72.2.9	spi_32bit_read_write	162
7.72.2.10	spi_64bit_read_write	162
7.72.2.11	warm_start_enable	162
7.72.2.12	restart_info_src	162
7.72.2.13	restart_info_port	163
7.72.2.14	mux_mode	163
7.72.2.15	pi	163
7.72.2.16	serdes	163
7.72.2.17	qs_conf	163
7.73	vtss_inst_create_t Struct Reference	164
7.73.1	Detailed Description	164
7.73.2	Field Documentation	164
7.73.2.1	target	164
7.74	vtss_ip_addr_t Struct Reference	164
7.74.1	Detailed Description	165
7.74.2	Field Documentation	165
7.74.2.1	type	165
7.74.2.2	ipv4	165
7.74.2.3	ipv6	165
7.74.2.4	addr	165
7.75	vtss_ip_network_t Struct Reference	166
7.75.1	Detailed Description	166
7.75.2	Field Documentation	166
7.75.2.1	address	166

7.75.2.2 <code>prefix_size</code>	166
7.76 <code>vtss_ipv4_network_t</code> Struct Reference	166
7.76.1 Detailed Description	167
7.76.2 Field Documentation	167
7.76.2.1 <code>address</code>	167
7.76.2.2 <code>prefix_size</code>	167
7.77 <code>vtss_ipv4_uc_t</code> Struct Reference	167
7.77.1 Detailed Description	168
7.77.2 Field Documentation	168
7.77.2.1 <code>network</code>	168
7.77.2.2 <code>destination</code>	168
7.78 <code>vtss_ipv6_network_t</code> Struct Reference	168
7.78.1 Detailed Description	169
7.78.2 Field Documentation	169
7.78.2.1 <code>address</code>	169
7.78.2.2 <code>prefix_size</code>	169
7.79 <code>vtss_ipv6_t</code> Struct Reference	169
7.79.1 Detailed Description	169
7.79.2 Field Documentation	170
7.79.2.1 <code>addr</code>	170
7.80 <code>vtss_ipv6_uc_t</code> Struct Reference	170
7.80.1 Detailed Description	170
7.80.2 Field Documentation	170
7.80.2.1 <code>network</code>	170
7.80.2.2 <code>destination</code>	171
7.81 <code>vtss_irq_conf_t</code> Struct Reference	171
7.81.1 Detailed Description	171
7.81.2 Field Documentation	171
7.81.2.1 <code>external</code>	171
7.81.2.2 <code>destination</code>	172

7.82 vtss_irq_status_t Struct Reference . . . . .	172
7.82.1 Detailed Description . . . . .	172
7.82.2 Field Documentation . . . . .	172
7.82.2.1 active . . . . .	172
7.82.2.2 raw_ident . . . . .	173
7.82.2.3 raw_status . . . . .	173
7.82.2.4 raw_mask . . . . .	173
7.83 vtss_l3_common_conf_t Struct Reference . . . . .	173
7.83.1 Detailed Description . . . . .	173
7.83.2 Field Documentation . . . . .	174
7.83.2.1 rleg_mode . . . . .	174
7.83.2.2 base_address . . . . .	174
7.83.2.3 routing_enable . . . . .	174
7.84 vtss_l3_counters_t Struct Reference . . . . .	174
7.84.1 Detailed Description . . . . .	175
7.84.2 Field Documentation . . . . .	175
7.84.2.1 ipv4uc_received_octets . . . . .	175
7.84.2.2 ipv4uc_received_frames . . . . .	175
7.84.2.3 ipv6uc_received_octets . . . . .	175
7.84.2.4 ipv6uc_received_frames . . . . .	175
7.84.2.5 ipv4uc_transmitted_octets . . . . .	176
7.84.2.6 ipv4uc_transmitted_frames . . . . .	176
7.84.2.7 ipv6uc_transmitted_octets . . . . .	176
7.84.2.8 ipv6uc_transmitted_frames . . . . .	176
7.85 vtss_l3_neighbour_t Struct Reference . . . . .	176
7.85.1 Detailed Description . . . . .	177
7.85.2 Field Documentation . . . . .	177
7.85.2.1 dmac . . . . .	177
7.85.2.2 vlan . . . . .	177
7.85.2.3 dip . . . . .	177

7.86 vtss_l3_rleg_conf_t Struct Reference . . . . .	178
7.86.1 Detailed Description . . . . .	178
7.86.2 Field Documentation . . . . .	178
7.86.2.1 ipv4_unicast_enable . . . . .	178
7.86.2.2 ipv6_unicast_enable . . . . .	178
7.86.2.3 ipv4_icmp_redirect_enable . . . . .	179
7.86.2.4 ipv6_icmp_redirect_enable . . . . .	179
7.86.2.5 vlan . . . . .	179
7.86.2.6 vrid0_enable . . . . .	179
7.86.2.7 vrid0 . . . . .	180
7.86.2.8 vrid1_enable . . . . .	180
7.86.2.9 vrid1 . . . . .	180
7.87 vtss_lcpll_status_t Struct Reference . . . . .	180
7.87.1 Detailed Description . . . . .	181
7.87.2 Field Documentation . . . . .	181
7.87.2.1 lock_status . . . . .	181
7.87.2.2 cal_done . . . . .	181
7.87.2.3 cal_error . . . . .	181
7.87.2.4 fsm_lock . . . . .	181
7.87.2.5 fsm_stat . . . . .	182
7.87.2.6 gain_stat . . . . .	182
7.88 vtss_learn_mode_t Struct Reference . . . . .	182
7.88.1 Detailed Description . . . . .	182
7.88.2 Field Documentation . . . . .	182
7.88.2.1 automatic . . . . .	183
7.88.2.2 cpu . . . . .	183
7.88.2.3 discard . . . . .	183
7.89 vtss_mac_t Struct Reference . . . . .	183
7.89.1 Detailed Description . . . . .	183
7.89.2 Field Documentation . . . . .	184

---

7.89.2.1  addr . . . . .	184
7.90  vtss_mac_table_entry_t Struct Reference . . . . .	184
7.90.1  Detailed Description . . . . .	184
7.90.2  Field Documentation . . . . .	184
7.90.2.1  vid_mac . . . . .	185
7.90.2.2  destination . . . . .	185
7.90.2.3  copy_to_cpu . . . . .	185
7.90.2.4  locked . . . . .	185
7.90.2.5  aged . . . . .	185
7.90.2.6  cpu_queue . . . . .	186
7.90.2.7  enable . . . . .	186
7.90.2.8  remote_entry . . . . .	186
7.90.2.9  upsid . . . . .	186
7.90.2.10  upspn . . . . .	186
7.90.2.11  vstax2 . . . . .	187
7.91  vtss_mac_table_status_t Struct Reference . . . . .	187
7.91.1  Detailed Description . . . . .	187
7.91.2  Field Documentation . . . . .	187
7.91.2.1  learned . . . . .	187
7.91.2.2  replaced . . . . .	188
7.91.2.3  moved . . . . .	188
7.91.2.4  aged . . . . .	188
7.92  vtss_mirror_conf_t Struct Reference . . . . .	188
7.92.1  Detailed Description . . . . .	189
7.92.2  Field Documentation . . . . .	189
7.92.2.1  port_no . . . . .	189
7.92.2.2  fwd_enable . . . . .	189
7.92.2.3  tag . . . . .	189
7.92.2.4  vid . . . . .	189
7.92.2.5  pcp . . . . .	190

---

7.92.2.6 dei . . . . .	190
7.93 vtss_mtimer_t Struct Reference . . . . .	190
7.93.1 Detailed Description . . . . .	190
7.93.2 Field Documentation . . . . .	190
7.93.2.1 timeout . . . . .	191
7.93.2.2 now . . . . .	191
7.94 vtss_npi_conf_t Struct Reference . . . . .	191
7.94.1 Detailed Description . . . . .	191
7.94.2 Field Documentation . . . . .	191
7.94.2.1 enable . . . . .	192
7.94.2.2 port_no . . . . .	192
7.95 vtss_os_timestamp_t Struct Reference . . . . .	192
7.95.1 Detailed Description . . . . .	192
7.95.2 Field Documentation . . . . .	192
7.95.2.1 hw_cnt . . . . .	193
7.96 vtss_packet_dma_conf_t Struct Reference . . . . .	193
7.96.1 Detailed Description . . . . .	193
7.96.2 Field Documentation . . . . .	193
7.96.2.1 dma_enable . . . . .	193
7.97 vtss_packet_frame_info_t Struct Reference . . . . .	194
7.97.1 Detailed Description . . . . .	194
7.97.2 Field Documentation . . . . .	194
7.97.2.1 port_no . . . . .	194
7.97.2.2 glag_no . . . . .	194
7.97.2.3 vid . . . . .	195
7.97.2.4 port_tx . . . . .	195
7.97.2.5 aggr_rx_disable . . . . .	195
7.97.2.6 aggr_tx_disable . . . . .	195
7.98 vtss_packet_port_filter_t Struct Reference . . . . .	195
7.98.1 Detailed Description . . . . .	196

---

7.98.2 Field Documentation . . . . .	196
7.98.2.1 filter . . . . .	196
7.98.2.2 tpid . . . . .	196
7.99 vtss_packet_port_info_t Struct Reference . . . . .	196
7.99.1 Detailed Description . . . . .	197
7.99.2 Field Documentation . . . . .	197
7.99.2.1 port_no . . . . .	197
7.99.2.2 glag_no . . . . .	197
7.99.2.3 vid . . . . .	197
7.99.2.4 aggr_rx_disable . . . . .	198
7.99.2.5 aggr_tx_disable . . . . .	198
7.100vtss_packet_rx_conf_t Struct Reference . . . . .	198
7.100.1 Detailed Description . . . . .	198
7.100.2 Field Documentation . . . . .	198
7.100.2.1 queue . . . . .	199
7.100.2.2 reg . . . . .	199
7.100.2.3 map . . . . .	199
7.100.2.4 grp_map . . . . .	199
7.101vtss_packet_rx_header_t Struct Reference . . . . .	199
7.101.1 Detailed Description . . . . .	200
7.101.2 Field Documentation . . . . .	200
7.101.2.1 length . . . . .	200
7.101.2.2 port_no . . . . .	200
7.101.2.3 queue_mask . . . . .	200
7.101.2.4 learn . . . . .	201
7.101.2.5 arrived_tagged . . . . .	201
7.101.2.6 tag . . . . .	201
7.101.2.7 vstax . . . . .	201
7.102vtss_packet_rx_info_t Struct Reference . . . . .	201
7.102.1 Detailed Description . . . . .	202

7.102.2 Field Documentation . . . . .	202
7.102.2.1 hints . . . . .	203
7.102.2.2 length . . . . .	203
7.102.2.3 port_no . . . . .	203
7.102.2.4 glag_no . . . . .	204
7.102.2.5 tag_type . . . . .	204
7.102.2.6 tag . . . . .	205
7.102.2.7 stripped_tag . . . . .	205
7.102.2.8 xtr_qu_mask . . . . .	206
7.102.2.9 cos . . . . .	206
7.102.2.10acl_hit . . . . .	206
7.102.2.11acl_idx . . . . .	207
7.102.2.12sw_tstamp . . . . .	207
7.102.2.13stamp_id . . . . .	207
7.102.2.14stamp_id_decoded . . . . .	208
7.102.2.15hw_tstamp . . . . .	208
7.102.2.16hw_tstamp_decoded . . . . .	208
7.102.2.17sflow_type . . . . .	209
7.102.2.18sflow_port_no . . . . .	209
7.102.2.19bam_info . . . . .	210
7.102.2.20bam_info_decoded . . . . .	210
7.102.2.21isidx . . . . .	210
7.102.2.22vstax . . . . .	211
7.103vtss_packet_rx_meta_t Struct Reference . . . . .	211
7.103.1 Detailed Description . . . . .	211
7.103.2 Field Documentation . . . . .	212
7.103.2.1 no_wait . . . . .	212
7.103.2.2 chip_no . . . . .	212
7.103.2.3 xtr_qu . . . . .	213
7.103.2.4 etype . . . . .	213

7.103.2.5 fcs . . . . .	214
7.103.2.6 sw_tstamp . . . . .	214
7.103.2.7 length . . . . .	215
7.104vtss_packet_rx_port_conf_t Struct Reference . . . . .	215
7.104.1 Detailed Description . . . . .	215
7.104.2 Field Documentation . . . . .	216
7.104.2.1 bpdu_reg . . . . .	216
7.104.2.2 garp_reg . . . . .	216
7.105vtss_packet_rx_queue_conf_t Struct Reference . . . . .	216
7.105.1 Detailed Description . . . . .	216
7.105.2 Field Documentation . . . . .	216
7.105.2.1 size . . . . .	217
7.105.2.2 npi . . . . .	217
7.106vtss_packet_rx_queue_map_t Struct Reference . . . . .	217
7.106.1 Detailed Description . . . . .	217
7.106.2 Field Documentation . . . . .	218
7.106.2.1 bpdu_queue . . . . .	218
7.106.2.2 garp_queue . . . . .	218
7.106.2.3 learn_queue . . . . .	218
7.106.2.4 igmp_queue . . . . .	218
7.106.2.5 ipmc_ctrl_queue . . . . .	218
7.106.2.6 mac_vid_queue . . . . .	219
7.106.2.7 stack_queue . . . . .	219
7.106.2.8 sflow_queue . . . . .	219
7.106.2.9 lrn_all_queue . . . . .	219
7.106.2.103_uc_queue . . . . .	219
7.106.2.113_other_queue . . . . .	220
7.107vtss_packet_rx_queue_npi_conf_t Struct Reference . . . . .	220
7.107.1 Detailed Description . . . . .	220
7.107.2 Field Documentation . . . . .	220

7.107.2.1 enable . . . . .	220
7.108vtss_packet_rx_reg_t Struct Reference . . . . .	221
7.108.1 Detailed Description . . . . .	221
7.108.2 Field Documentation . . . . .	221
7.108.2.1 bpdu_cpu_only . . . . .	221
7.108.2.2 garp_cpu_only . . . . .	221
7.108.2.3 ipmc_ctrl_cpu_copy . . . . .	222
7.108.2.4 igmp_cpu_only . . . . .	222
7.108.2.5 mld_cpu_only . . . . .	222
7.109vtss_packet_tx_ifh_t Struct Reference . . . . .	222
7.109.1 Detailed Description . . . . .	222
7.109.2 Field Documentation . . . . .	223
7.109.2.1 length . . . . .	223
7.109.2.2 ifh . . . . .	223
7.110vtss_packet_tx_info_t Struct Reference . . . . .	223
7.110.1 Detailed Description . . . . .	224
7.110.2 Field Documentation . . . . .	224
7.110.2.1 switch_frm . . . . .	224
7.110.2.2 dst_port_mask . . . . .	225
7.110.2.3 frm_len . . . . .	225
7.110.2.4 tag . . . . .	226
7.110.2.5 aggr_code . . . . .	226
7.110.2.6 cos . . . . .	227
7.110.2.7 tx_vstax_hdr . . . . .	228
7.110.2.8 bin . . . . .	228
7.110.2.9 sym . . . . .	229
7.110.2.10vstax . . . . .	229
7.110.2.11ptp_action . . . . .	229
7.110.2.12ptp_id . . . . .	230
7.110.2.13ptp_timestamp . . . . .	230

7.110.2.14	atch_timestamp	231
7.110.2.15	am_type	231
7.110.2.16	sdx	232
7.110.2.17	sdx_dont_use	232
7.110.2.18	dp	233
7.110.2.19	masquerade_port	233
7.110.2.20	pdu_offset	234
7.111	vtss_phy_10g_apc_conf_t Struct Reference	234
7.111.1	Detailed Description	234
7.111.2	Field Documentation	234
7.111.2.1	op_mode	235
7.111.2.2	op_mode_flag	235
7.112	vtss_phy_10g_apc_status_t Struct Reference	235
7.112.1	Detailed Description	235
7.112.2	Field Documentation	235
7.112.2.1	reset	236
7.112.2.2	freeze	236
7.113	vtss_phy_10g_auto_failover_conf_t Struct Reference	236
7.113.1	Detailed Description	236
7.113.2	Field Documentation	237
7.113.2.1	port_no	237
7.113.2.2	evnt	237
7.113.2.3	trig_ch_id	237
7.113.2.4	is_host_side	237
7.113.2.5	channel_id	237
7.113.2.6	v_gpio	238
7.113.2.7	a_gpio	238
7.113.2.8	enable	238
7.113.2.9	filter	238
7.113.2.10	ltr_val	238

7.114vtss_phy_10g_base_kr_autoneg_t Struct Reference . . . . .	239
7.114.1 Detailed Description . . . . .	239
7.114.2 Field Documentation . . . . .	239
7.114.2.1 an_restart . . . . .	239
7.114.2.2 an_enable . . . . .	239
7.114.2.3 an_reset . . . . .	240
7.115vtss_phy_10g_base_kr_conf_t Struct Reference . . . . .	240
7.115.1 Detailed Description . . . . .	240
7.115.2 Field Documentation . . . . .	240
7.115.2.1 cm1 . . . . .	240
7.115.2.2 c0 . . . . .	241
7.115.2.3 c1 . . . . .	241
7.115.2.4 ampl . . . . .	241
7.115.2.5 slewrate . . . . .	241
7.115.2.6 en_ob . . . . .	241
7.115.2.7 ser_inv . . . . .	242
7.116vtss_phy_10g_base_kr_ld_adv_abil_t Struct Reference . . . . .	242
7.116.1 Detailed Description . . . . .	242
7.116.2 Field Documentation . . . . .	242
7.116.2.1 adv_1g . . . . .	242
7.116.2.2 adv_10g . . . . .	243
7.116.2.3 fec_abil . . . . .	243
7.116.2.4 fec_req . . . . .	243
7.117vtss_phy_10g_base_kr_status_t Struct Reference . . . . .	243
7.117.1 Detailed Description . . . . .	243
7.117.2 Field Documentation . . . . .	244
7.117.2.1 aneg . . . . .	244
7.117.2.2 train . . . . .	244
7.117.2.3 fec . . . . .	244
7.118vtss_phy_10g_base_kr_train_aneg_t Struct Reference . . . . .	244

---

7.118.1 Detailed Description	245
7.118.2 Field Documentation	245
7.118.2.1 training	245
7.118.2.2 autoneg	245
7.118.2.3 ld_abil	245
7.118.2.4 host_kr	245
7.118.2.5 line_kr	246
7.119vtss_phy_10g_base_kr_training_t Struct Reference	246
7.119.1 Detailed Description	246
7.119.2 Field Documentation	246
7.119.2.1 enable	246
7.119.2.2 trmthd_cp	247
7.119.2.3 trmthd_c0	247
7.119.2.4 trmthd_cm	247
7.119.2.5 ld_pre_init	247
7.120vtss_phy_10g_ckout_conf_t Struct Reference	247
7.120.1 Detailed Description	248
7.120.2 Field Documentation	248
7.120.2.1 mode	248
7.120.2.2 src	248
7.120.2.3 freq	248
7.120.2.4 squelch_inv	249
7.120.2.5 enable	249
7.120.2.6 ckout_sel	249
7.121vtss_phy_10g_clause_37_adv_t Struct Reference	249
7.121.1 Detailed Description	250
7.121.2 Field Documentation	250
7.121.2.1 fdx	250
7.121.2.2 hdx	250
7.121.2.3 symmetric_pause	250

7.121.2.4 asymmetric_pause . . . . .	250
7.121.2.5 remote_fault . . . . .	251
7.121.2.6 acknowledge . . . . .	251
7.121.2.7 next_page . . . . .	251
7.122vtss_phy_10g_clause_37_cmn_status_t Struct Reference . . . . .	251
7.122.1 Detailed Description . . . . .	251
7.122.2 Field Documentation . . . . .	252
7.122.2.1 line . . . . .	252
7.122.2.2 host . . . . .	252
7.123vtss_phy_10g_clause_37_control_t Struct Reference . . . . .	252
7.123.1 Detailed Description . . . . .	252
7.123.2 Field Documentation . . . . .	253
7.123.2.1 enable . . . . .	253
7.123.2.2 advertisement . . . . .	253
7.123.2.3 enable_pass_thru . . . . .	253
7.123.2.4 line . . . . .	253
7.123.2.5 host . . . . .	253
7.123.2.6 l_h . . . . .	254
7.124vtss_phy_10g_clause_37_status_t Struct Reference . . . . .	254
7.124.1 Detailed Description . . . . .	254
7.124.2 Field Documentation . . . . .	254
7.124.2.1 link . . . . .	254
7.124.2.2 complete . . . . .	255
7.124.2.3 partner_advertisement . . . . .	255
7.124.2.4 autoneg . . . . .	255
7.125vtss_phy_10g_clk_src_t Struct Reference . . . . .	255
7.125.1 Detailed Description . . . . .	255
7.125.2 Field Documentation . . . . .	256
7.125.2.1 is_high_amp . . . . .	256
7.126vtss_phy_10g_cnt_t Struct Reference . . . . .	256

7.126.1 Detailed Description	256
7.126.2 Field Documentation	256
7.126.2.1 <code>pcs</code>	256
7.127vtss_phy_10g_fifo_sync_t Struct Reference	257
7.127.1 Detailed Description	257
7.127.2 Field Documentation	257
7.127.2.1 <code>bypass_in_api</code>	257
7.127.2.2 <code>skip_rev_check</code>	257
7.127.2.3 <code>pr</code>	258
7.128vtss_phy_10g_fw_status_t Struct Reference	258
7.128.1 Detailed Description	258
7.128.2 Field Documentation	258
7.128.2.1 <code>edc_fw_rev</code>	258
7.128.2.2 <code>edc_fw_chksum</code>	259
7.128.2.3 <code>icpu_activity</code>	259
7.128.2.4 <code>edc_fw_api_load</code>	259
7.129vtss_phy_10g_host_clk_conf_t Struct Reference	259
7.129.1 Detailed Description	260
7.129.2 Field Documentation	260
7.129.2.1 <code>mode</code>	260
7.129.2.2 <code>recvrd_clk_sel</code>	260
7.129.2.3 <code>clk_sel_no</code>	260
7.130vtss_phy_10g_i2c_slave_conf_t Struct Reference	260
7.130.1 Detailed Description	261
7.130.2 Field Documentation	261
7.130.2.1 <code>slave_id</code>	261
7.130.2.2 <code>prescale</code>	261
7.131vtss_phy_10g_ib_conf_t Struct Reference	261
7.131.1 Detailed Description	262
7.131.2 Field Documentation	262

7.131.2.1 offs . . . . .	262
7.131.2.2 gain . . . . .	262
7.131.2.3 gainadj . . . . .	263
7.131.2.4 l . . . . .	263
7.131.2.5 c . . . . .	263
7.131.2.6 agc . . . . .	263
7.131.2.7 dfe1 . . . . .	263
7.131.2.8 dfe2 . . . . .	264
7.131.2.9 dfe3 . . . . .	264
7.131.2.10dfe4 . . . . .	264
7.131.2.11ld . . . . .	264
7.131.2.12prbs . . . . .	264
7.131.2.13prbs_inv . . . . .	265
7.131.2.14apc_bit_mask . . . . .	265
7.131.2.15freeze_bit_mask . . . . .	265
7.131.2.16config_bit_mask . . . . .	265
7.131.2.17s_host . . . . .	265
7.132vtss_phy_10g_ib_status_t Struct Reference . . . . .	266
7.132.1 Detailed Description . . . . .	266
7.132.2 Field Documentation . . . . .	266
7.132.2.1 ib_conf . . . . .	266
7.132.2.2 sig_det . . . . .	266
7.132.2.3 bit_errors . . . . .	267
7.133vtss_phy_10g_ib_storage_t Struct Reference . . . . .	267
7.133.1 Detailed Description . . . . .	267
7.133.2 Field Documentation . . . . .	267
7.133.2.1 ib_storage_bool . . . . .	267
7.133.2.2 ib_storage . . . . .	268
7.134vtss_phy_10g_id_t Struct Reference . . . . .	268
7.134.1 Detailed Description . . . . .	268

---

7.134.2 Field Documentation . . . . .	268
7.134.2.1 part_number . . . . .	268
7.134.2.2 revision . . . . .	269
7.134.2.3 channel_id . . . . .	269
7.134.2.4 family . . . . .	269
7.134.2.5 type . . . . .	269
7.134.2.6 phy_api_base_no . . . . .	269
7.134.2.7 device_feature_status . . . . .	270
7.135vtss_phy_10g_init_parm_t Struct Reference . . . . .	270
7.135.1 Detailed Description . . . . .	270
7.135.2 Field Documentation . . . . .	270
7.135.2.1 channel_conf . . . . .	270
7.136vtss_phy_10g_jitter_conf_t Struct Reference . . . . .	271
7.136.1 Detailed Description . . . . .	271
7.136.2 Field Documentation . . . . .	271
7.136.2.1 incr_levn . . . . .	271
7.136.2.2 levn . . . . .	271
7.136.2.3 vtail . . . . .	272
7.137vtss_phy_10g_kr_status_aneg_t Struct Reference . . . . .	272
7.137.1 Detailed Description . . . . .	272
7.137.2 Field Documentation . . . . .	272
7.137.2.1 complete . . . . .	272
7.137.2.2 active . . . . .	273
7.137.2.3 request_10g . . . . .	273
7.137.2.4 request_1g . . . . .	273
7.137.2.5 request_fec_change . . . . .	273
7.137.2.6 fec_enable . . . . .	273
7.137.2.7 sm . . . . .	274
7.137.2.8 lp_aneg_able . . . . .	274
7.137.2.9 block_lock . . . . .	274

7.138vtss_phy_10g_kr_status_fec_t Struct Reference . . . . .	274
7.138.1 Detailed Description . . . . .	274
7.138.2 Field Documentation . . . . .	275
7.138.2.1 enable . . . . .	275
7.138.2.2 corrected_block_cnt . . . . .	275
7.138.2.3 uncorrected_block_cnt . . . . .	275
7.139vtss_phy_10g_kr_status_train_t Struct Reference . . . . .	275
7.139.1 Detailed Description . . . . .	276
7.139.2 Field Documentation . . . . .	276
7.139.2.1 complete . . . . .	276
7.139.2.2 cm_ob_tap_result . . . . .	276
7.139.2.3 cp_ob_tap_result . . . . .	276
7.139.2.4 c0_ob_tap_result . . . . .	276
7.140vtss_phy_10g_lane_sync_conf_t Struct Reference . . . . .	277
7.140.1 Detailed Description . . . . .	277
7.140.2 Field Documentation . . . . .	277
7.140.2.1 enable . . . . .	277
7.140.2.2 tx_macro . . . . .	277
7.140.2.3 rx_macro . . . . .	278
7.140.2.4 rx_ch . . . . .	278
7.140.2.5 tx_ch . . . . .	278
7.141vtss_phy_10g_line_clk_conf_t Struct Reference . . . . .	278
7.141.1 Detailed Description . . . . .	279
7.141.2 Field Documentation . . . . .	279
7.141.2.1 mode . . . . .	279
7.141.2.2 recvrd_clk_sel . . . . .	279
7.141.2.3 clk_sel_no . . . . .	279
7.142vtss_phy_10g_loopback_t Struct Reference . . . . .	279
7.142.1 Detailed Description . . . . .	280
7.142.2 Field Documentation . . . . .	280

---

7.142.2.1 <code>lb_type</code>	280
7.142.2.2 <code>enable</code>	280
7.143vtss_phy_10g_mode_t Struct Reference	280
7.143.1 Detailed Description	282
7.143.2 Member Enumeration Documentation	282
7.143.2.1 anonymous enum	282
7.143.3 Field Documentation	282
7.143.3.1 <code>oper_mode</code>	282
7.143.3.2 <code>interface</code>	282
7.143.3.3 <code>wrefclk</code>	283
7.143.3.4 <code>high_input_gain</code>	283
7.143.3.5 <code>xfi_pol_invert</code>	283
7.143.3.6 <code>xaui_lane_flip</code>	283
7.143.3.7 <code>channel_id</code>	283
7.143.3.8 <code>hl_clk_synth</code>	284
7.143.3.9 <code>rcvrd_clk</code>	284
7.143.3.10 <code>rcvrd_clk_div</code>	284
7.143.3.11 <code>sref_clk_div</code>	284
7.143.3.12 <code>wref_clk_div</code>	284
7.143.3.13 <code>edc_fw_load</code>	285
7.143.3.14 <code>use_conf</code>	285
7.143.3.15 <code>ob_conf</code>	285
7.143.3.16 <code>b_conf</code>	285
7.143.3.17 <code>dig_offset_reg</code>	285
7.143.3.18 <code>apc_offs_ctrl</code>	286
7.143.3.19 <code>apc_line_id_ctrl</code>	286
7.143.3.20 <code>apc_host_id_ctrl</code>	286
7.143.3.21 <code>d_filter</code>	286
7.143.3.22 <code>cfg0</code>	286
7.143.3.23 <code>b_ini_ip</code>	287

7.143.3.24b_min_lp . . . . .	287
7.143.3.25b_max_lp . . . . .	287
7.143.3.26apc_eqz_offs_par_cfg . . . . .	287
7.143.3.27apc_line_eqz_id_ctrl . . . . .	287
7.143.3.28apc_host_eqz_id_ctrl . . . . .	288
7.143.3.29_offset_guard . . . . .	288
7.143.3.30h_offset_guard . . . . .	288
7.143.3.31serdes_conf . . . . .	288
7.143.3.32apc_ib_regulator . . . . .	288
7.143.3.33pma_txratecontrol . . . . .	289
7.143.3.34venice_rev_a_los_detection_workaround . . . . .	289
7.143.3.35ddr_mode . . . . .	289
7.143.3.36master . . . . .	289
7.143.3.37rate . . . . .	289
7.143.3.38polarity . . . . .	290
7.143.3.39s_host_wan . . . . .	290
7.143.3.40h_clk_src . . . . .	290
7.143.3.41l_clk_src . . . . .	290
7.143.3.42ref_for_host . . . . .	290
7.143.3.43ink_6g_distance . . . . .	291
7.143.3.44h_media . . . . .	291
7.143.3.45_media . . . . .	291
7.143.3.46h_ib_conf . . . . .	291
7.143.3.47_ib_conf . . . . .	291
7.143.3.48h_apc_conf . . . . .	292
7.143.3.49_apc_conf . . . . .	292
7.143.3.50enable_pass_thru . . . . .	292
7.143.3.51is_init . . . . .	292
7.143.3.52sd6g_calib_done . . . . .	292
7.144vtss_phy_10g_ob_status_t Struct Reference . . . . .	293

7.144.1 Detailed Description . . . . .	293
7.144.2 Field Documentation . . . . .	293
7.144.2.1 r_ctrl . . . . .	293
7.144.2.2 c_ctrl . . . . .	293
7.144.2.3 slew . . . . .	294
7.144.2.4 levn . . . . .	294
7.144.2.5 d_fltr . . . . .	294
7.144.2.6 v3 . . . . .	294
7.144.2.7 vp . . . . .	294
7.144.2.8 v4 . . . . .	295
7.144.2.9 v5 . . . . .	295
7.144.2.10s_host . . . . .	295
7.145vtss_phy_10g_pcs_prbs_gen_conf_t Struct Reference . . . . .	295
7.145.1 Detailed Description . . . . .	295
7.145.2 Field Documentation . . . . .	296
7.145.2.1 prbs_gen . . . . .	296
7.146vtss_phy_10g_pcs_prbs_mon_conf_t Struct Reference . . . . .	296
7.146.1 Detailed Description . . . . .	296
7.146.2 Field Documentation . . . . .	296
7.146.2.1 prbs_mon . . . . .	296
7.146.2.2 error_counter . . . . .	297
7.147vtss_phy_10g_pkt_gen_conf_t Struct Reference . . . . .	297
7.147.1 Detailed Description . . . . .	297
7.147.2 Field Documentation . . . . .	297
7.147.2.1 enable . . . . .	298
7.147.2.2 ptp . . . . .	298
7.147.2.3 ingress . . . . .	298
7.147.2.4 frames . . . . .	298
7.147.2.5 frame_single . . . . .	298
7.147.2.6 etype . . . . .	299

7.147.2.7 pkt_len . . . . .	299
7.147.2.8 ipg_len . . . . .	299
7.147.2.9 smac . . . . .	299
7.147.2.10dmac . . . . .	299
7.147.2.11ptp_ts_sec . . . . .	300
7.147.2.12ptp_ts_ns . . . . .	300
7.147.2.13rate . . . . .	300
7.148vtss_phy_10g_pkt_mon_conf_t Struct Reference . . . . .	300
7.148.1 Detailed Description . . . . .	301
7.148.2 Field Documentation . . . . .	301
7.148.2.1 enable . . . . .	301
7.148.2.2 update . . . . .	301
7.148.2.3 reset . . . . .	301
7.148.2.4 good_crc . . . . .	301
7.148.2.5 bad_crc . . . . .	302
7.148.2.6 frag . . . . .	302
7.148.2.7 lfault . . . . .	302
7.148.2.8 ber . . . . .	302
7.149vtss_phy_10g_polarity_inv_t Struct Reference . . . . .	302
7.149.1 Detailed Description . . . . .	303
7.149.2 Field Documentation . . . . .	303
7.149.2.1 line_rx . . . . .	303
7.149.2.2 line_tx . . . . .	303
7.149.2.3 host_rx . . . . .	303
7.149.2.4 host_tx . . . . .	304
7.150vtss_phy_10g_prbs_gen_conf_t Struct Reference . . . . .	304
7.150.1 Detailed Description . . . . .	304
7.150.2 Field Documentation . . . . .	304
7.150.2.1 enable . . . . .	304
7.150.2.2 prbsn_tx_sel . . . . .	305

---

7.150.2.3 <code>line</code>	305
7.150.2.4 <code>prbsn_tx_io</code>	305
7.150.2.5 <code>prbsn_tx_iw</code>	305
7.151vtss_phy_10g_prbs_mon_conf_t Struct Reference	305
7.151.1 Detailed Description	306
7.151.2 Field Documentation	306
7.151.2.1 <code>enable</code>	306
7.151.2.2 <code>line</code>	306
7.151.2.3 <code>max_bist_frames</code>	307
7.151.2.4 <code>error_states</code>	307
7.151.2.5 <code>des_interface_width</code>	307
7.151.2.6 <code>prbsn_sel</code>	307
7.151.2.7 <code>prbs_check_input_invert</code>	307
7.151.2.8 <code>no_of_errors</code>	308
7.151.2.9 <code>bist_mode</code>	308
7.151.2.10 <code>error_status</code>	308
7.151.2.11 <code>PRBS_status</code>	308
7.151.2.12 <code>main_status</code>	308
7.151.2.13 <code>stuck_at_par</code>	309
7.151.2.14 <code>stuck_at_01</code>	309
7.151.2.15 <code>no_sync</code>	309
7.151.2.16 <code>instable</code>	309
7.151.2.17 <code>incomplete</code>	309
7.151.2.18 <code>active</code>	310
7.152vtss_phy_10g_rxckout_conf_t Struct Reference	310
7.152.1 Detailed Description	310
7.152.2 Field Documentation	310
7.152.2.1 <code>mode</code>	310
7.152.2.2 <code>squelch_on_pcs_fault</code>	311
7.152.2.3 <code>squelch_on_lopc</code>	311

7.153vtss_phy_10g_sckout_conf_t Struct Reference . . . . .	311
7.153.1 Detailed Description . . . . .	311
7.153.2 Field Documentation . . . . .	311
7.153.2.1 mode . . . . .	312
7.153.2.2 src . . . . .	312
7.153.2.3 freq . . . . .	312
7.153.2.4 squelch_inv . . . . .	312
7.153.2.5 enable . . . . .	312
7.154vtss_phy_10g_serdes_status_t Struct Reference . . . . .	313
7.154.1 Detailed Description . . . . .	313
7.154.2 Field Documentation . . . . .	313
7.154.2.1 rcomp . . . . .	314
7.154.2.2 h_pll5g_lock_status . . . . .	314
7.154.2.3 h_pll5g_fsm_lock . . . . .	314
7.154.2.4 h_pll5g_fsm_stat . . . . .	314
7.154.2.5 h_pll5g_gain . . . . .	314
7.154.2.6 l_pll5g_lock_status . . . . .	315
7.154.2.7 l_pll5g_fsm_lock . . . . .	315
7.154.2.8 l_pll5g_fsm_stat . . . . .	315
7.154.2.9 l_pll5g_gain . . . . .	315
7.154.2.10h_rx_rcpll_lock_status . . . . .	315
7.154.2.11h_rx_rcpll_range . . . . .	316
7.154.2.12h_rx_rcpll_vco_load . . . . .	316
7.154.2.13h_rx_rcpll_fsm_status . . . . .	316
7.154.2.14_rx_rcpll_lock_status . . . . .	316
7.154.2.15_rx_rcpll_range . . . . .	316
7.154.2.16_rx_rcpll_vco_load . . . . .	317
7.154.2.17_rx_rcpll_fsm_status . . . . .	317
7.154.2.18h_tx_rcpll_lock_status . . . . .	317
7.154.2.19h_tx_rcpll_range . . . . .	317

7.154.2.20_tx_rcpll_vco_load . . . . .	317
7.154.2.21_tx_rcpll_fsm_status . . . . .	318
7.154.2.22_tx_rcpll_lock_status . . . . .	318
7.154.2.23_tx_rcpll_range . . . . .	318
7.154.2.24_tx_rcpll_vco_load . . . . .	318
7.154.2.25_tx_rcpll_fsm_status . . . . .	318
7.154.2.26_tx_pma . . . . .	319
7.154.2.27_tx_pcs . . . . .	319
7.154.2.28_pma . . . . .	319
7.154.2.29_pcs . . . . .	319
7.154.2.30_wis . . . . .	319
7.155vtss_phy_10g_srefclk_mode_t Struct Reference . . . . .	320
7.155.1 Detailed Description . . . . .	320
7.155.2 Field Documentation . . . . .	320
7.155.2.1 enable . . . . .	320
7.155.2.2 freq . . . . .	320
7.156vtss_phy_10g_status_t Struct Reference . . . . .	320
7.156.1 Detailed Description . . . . .	321
7.156.2 Field Documentation . . . . .	321
7.156.2.1 pma . . . . .	321
7.156.2.2 hpma . . . . .	321
7.156.2.3 wis . . . . .	321
7.156.2.4 pcs . . . . .	322
7.156.2.5 hpcs . . . . .	322
7.156.2.6 xs . . . . .	322
7.156.2.7 lpcs_1g . . . . .	322
7.156.2.8 hpcs_1g . . . . .	322
7.156.2.9 status . . . . .	323
7.156.2.10block_lock . . . . .	323
7.156.2.11lopc_stat . . . . .	323

7.157vtss_phy_10g_timestamp_val_t Struct Reference . . . . .	323
7.157.1 Detailed Description . . . . .	323
7.157.2 Field Documentation . . . . .	324
7.157.2.1 timestamp . . . . .	324
7.158vtss_phy_10g_txckout_conf_t Struct Reference . . . . .	324
7.158.1 Detailed Description . . . . .	324
7.158.2 Field Documentation . . . . .	324
7.158.2.1 mode . . . . .	324
7.159vtss_phy_10g_vscope_conf_t Struct Reference . . . . .	325
7.159.1 Detailed Description . . . . .	325
7.159.2 Field Documentation . . . . .	325
7.159.2.1 scan_type . . . . .	325
7.159.2.2 line . . . . .	325
7.159.2.3 enable . . . . .	325
7.159.2.4 error_thres . . . . .	326
7.160vtss_phy_10g_vscope_scan_conf_t Struct Reference . . . . .	326
7.160.1 Detailed Description . . . . .	326
7.160.2 Field Documentation . . . . .	326
7.160.2.1 line . . . . .	326
7.160.2.2 x_start . . . . .	327
7.160.2.3 y_start . . . . .	327
7.160.2.4 x_incr . . . . .	327
7.160.2.5 y_incr . . . . .	327
7.160.2.6 x_count . . . . .	327
7.160.2.7 y_count . . . . .	328
7.160.2.8 ber . . . . .	328
7.161vtss_phy_10g_vscope_scan_status_t Struct Reference . . . . .	328
7.161.1 Detailed Description . . . . .	328
7.161.2 Field Documentation . . . . .	328
7.161.2.1 scan_conf . . . . .	329

---

---

7.161.2.2 error_free_x . . . . .	329
7.161.2.3 error_free_y . . . . .	329
7.161.2.4 amp_range . . . . .	329
7.161.2.5 errors . . . . .	329
7.162vtss_phy_aneg_t Struct Reference . . . . .	330
7.162.1 Detailed Description . . . . .	330
7.162.2 Field Documentation . . . . .	330
7.162.2.1 speed_10m_hdx . . . . .	330
7.162.2.2 speed_10m_fdx . . . . .	330
7.162.2.3 speed_100m_hdx . . . . .	331
7.162.2.4 speed_100m_fdx . . . . .	331
7.162.2.5 speed_1g_fdx . . . . .	331
7.162.2.6 speed_1g_hdx . . . . .	331
7.162.2.7 symmetric_pause . . . . .	331
7.162.2.8 asymmetric_pause . . . . .	332
7.162.2.9 tx_remote_fault . . . . .	332
7.163vtss_phy_clock_conf_t Struct Reference . . . . .	332
7.163.1 Detailed Description . . . . .	332
7.163.2 Field Documentation . . . . .	332
7.163.2.1 src . . . . .	333
7.163.2.2 freq . . . . .	333
7.163.2.3 squelch . . . . .	333
7.164vtss_phy_conf_1g_t Struct Reference . . . . .	333
7.164.1 Detailed Description . . . . .	334
7.164.2 Field Documentation . . . . .	334
7.164.2.1 cfg . . . . .	334
7.164.2.2 val . . . . .	334
7.164.2.3 master . . . . .	334
7.165vtss_phy_conf_t Struct Reference . . . . .	334
7.165.1 Detailed Description . . . . .	335

---

7.165.2 Field Documentation . . . . .	335
7.165.2.1 mode . . . . .	335
7.165.2.2 forced . . . . .	335
7.165.2.3 aneg . . . . .	335
7.165.2.4 mdi . . . . .	336
7.165.2.5 flf . . . . .	336
7.165.2.6 sigdet . . . . .	336
7.165.2.7 unidir . . . . .	336
7.165.2.8 mac_if_pcs . . . . .	336
7.165.2.9 media_if_pcs . . . . .	337
7.165.2.10 force_ams_sel . . . . .	337
7.165.2.11 skip_coma . . . . .	337
7.166vtss_phy_daisy_chain_conf_t Struct Reference . . . . .	337
7.166.1 Detailed Description . . . . .	337
7.166.2 Field Documentation . . . . .	338
7.166.2.1 spi_daisy_input . . . . .	338
7.166.2.2 spi_daisy_output . . . . .	338
7.167vtss_phy_eee_conf_t Struct Reference . . . . .	338
7.167.1 Detailed Description . . . . .	338
7.167.2 Field Documentation . . . . .	338
7.167.2.1 eee_mode . . . . .	339
7.167.2.2 eee_ena_phy . . . . .	339
7.168vtss_phy_enhanced_led_control_t Struct Reference . . . . .	339
7.168.1 Detailed Description . . . . .	339
7.168.2 Field Documentation . . . . .	339
7.168.2.1 ser_led_output_1 . . . . .	340
7.168.2.2 ser_led_output_2 . . . . .	340
7.168.2.3 ser_led_frame_rate . . . . .	340
7.168.2.4 ser_led_select . . . . .	340
7.169vtss_phy_forced_t Struct Reference . . . . .	340

---

7.169.1 Detailed Description . . . . .	341
7.169.2 Field Documentation . . . . .	341
7.169.2.1 speed . . . . .	341
7.169.2.2 fdx . . . . .	341
7.170vtss_phy_led_mode_select_t Struct Reference . . . . .	341
7.170.1 Detailed Description . . . . .	342
7.170.2 Field Documentation . . . . .	342
7.170.2.1 mode . . . . .	342
7.170.2.2 number . . . . .	342
7.171vtss_phy_loopback_t Struct Reference . . . . .	342
7.171.1 Detailed Description . . . . .	343
7.171.2 Field Documentation . . . . .	343
7.171.2.1 far_end_enable . . . . .	343
7.171.2.2 near_end_enable . . . . .	343
7.171.2.3 connector_enable . . . . .	343
7.171.2.4 mac_serdes_input_enable . . . . .	343
7.171.2.5 mac_serdes_facility_enable . . . . .	344
7.171.2.6 mac_serdes_equipment_enable . . . . .	344
7.171.2.7 media_serdes_input_enable . . . . .	344
7.171.2.8 media_serdes_facility_enable . . . . .	344
7.171.2.9 media_serdes_equipment_enable . . . . .	344
7.172vtss_phy_ltc_freq_synth_s Struct Reference . . . . .	345
7.172.1 Detailed Description . . . . .	345
7.172.2 Field Documentation . . . . .	345
7.172.2.1 enable . . . . .	345
7.172.2.2 high_duty_cycle . . . . .	345
7.172.2.3 low_duty_cycle . . . . .	346
7.173vtss_phy_mac_serdes_ctrl_t Struct Reference . . . . .	346
7.173.1 Detailed Description . . . . .	346
7.173.2 Field Documentation . . . . .	346

---

7.173.2.1 disable . . . . .	347
7.173.2.2 restart . . . . .	347
7.173.2.3 pd_enable . . . . .	347
7.173.2.4 aneg_restart . . . . .	347
7.173.2.5 force_adv_ability . . . . .	347
7.173.2.6 sgmii_in_pre . . . . .	348
7.173.2.7 sgmii_out_pre . . . . .	348
7.173.2.8 serdes_aneg_ena . . . . .	348
7.173.2.9 serdes_pol_inv_in . . . . .	348
7.173.2.10 serdes_pol_inv_out . . . . .	348
7.173.2.11 fast_link_stat_ena . . . . .	349
7.173.2.12 inhibit_odd_start . . . . .	349
7.174 vtss_phy_media_serd_pcs_cntl_t Struct Reference . . . . .	349
7.174.1 Detailed Description . . . . .	349
7.174.2 Field Documentation . . . . .	350
7.174.2.1 remote_fault . . . . .	350
7.174.2.2 aneg_pd_detect . . . . .	350
7.174.2.3 force_adv_ability . . . . .	350
7.174.2.4 serdes_pol_inv_in . . . . .	350
7.174.2.5 serdes_pol_inv_out . . . . .	350
7.174.2.6 inhibit_odd_start . . . . .	351
7.174.2.7 force_hls . . . . .	351
7.174.2.8 force_fefi . . . . .	351
7.174.2.9 force_fefi_value . . . . .	351
7.175 vtss_phy_pcs_cnt_t Struct Reference . . . . .	351
7.175.1 Detailed Description . . . . .	352
7.175.2 Field Documentation . . . . .	352
7.175.2.1 block_lock_latched . . . . .	352
7.175.2.2 high_ber_latched . . . . .	352
7.175.2.3 ber_cnt . . . . .	352

7.175.2.4 err_blk_cnt . . . . .	353
7.176vtss_phy_power_conf_t Struct Reference . . . . .	353
7.176.1 Detailed Description . . . . .	353
7.176.2 Field Documentation . . . . .	353
7.176.2.1 mode . . . . .	353
7.177vtss_phy_power_status_t Struct Reference . . . . .	354
7.177.1 Detailed Description . . . . .	354
7.177.2 Field Documentation . . . . .	354
7.177.2.1 level . . . . .	354
7.178vtss_phy_reset_conf_t Struct Reference . . . . .	354
7.178.1 Detailed Description . . . . .	355
7.178.2 Field Documentation . . . . .	355
7.178.2.1 mac_if . . . . .	355
7.178.2.2 media_if . . . . .	355
7.178.2.3 rgmii . . . . .	355
7.178.2.4 tbi . . . . .	355
7.178.2.5 force . . . . .	356
7.178.2.6 pkt_mode . . . . .	356
7.178.2.7 i_cpu_en . . . . .	356
7.179vtss_phy_rgmii_conf_t Struct Reference . . . . .	356
7.179.1 Detailed Description . . . . .	356
7.179.2 Field Documentation . . . . .	357
7.179.2.1 rx_clk_skew_ps . . . . .	357
7.179.2.2 tx_clk_skew_ps . . . . .	357
7.180vtss_phy_statistic_t Struct Reference . . . . .	357
7.180.1 Detailed Description . . . . .	357
7.180.2 Field Documentation . . . . .	358
7.180.2.1 cu_good . . . . .	358
7.180.2.2 cu_bad . . . . .	358
7.180.2.3 serdes_tx_good . . . . .	358

7.180.2.4 serdes_tx_bad . . . . .	358
7.180.2.5 rx_err_cnt_base_tx . . . . .	359
7.180.2.6 media_mac_serdes_good . . . . .	359
7.180.2.7 media_mac_serdes_crc . . . . .	359
7.181vtss_phy_status_1g_t Struct Reference . . . . .	359
7.181.1 Detailed Description . . . . .	360
7.181.2 Field Documentation . . . . .	360
7.181.2.1 master_cfg_fault . . . . .	360
7.181.2.2 master . . . . .	360
7.182vtss_phy_tbi_conf_t Struct Reference . . . . .	360
7.182.1 Detailed Description . . . . .	360
7.182.2 Field Documentation . . . . .	361
7.182.2.1 aneg_enable . . . . .	361
7.183vtss_phy_timestamp_t Struct Reference . . . . .	361
7.183.1 Detailed Description . . . . .	361
7.183.2 Field Documentation . . . . .	361
7.183.2.1 high . . . . .	361
7.183.2.2 low . . . . .	362
7.183.2.3 seconds . . . . .	362
7.183.2.4 nanoseconds . . . . .	362
7.184vtss_phy_ts_ach_conf_t Struct Reference . . . . .	362
7.184.1 Detailed Description . . . . .	363
7.184.2 Field Documentation . . . . .	363
7.184.2.1 value [1/2] . . . . .	363
7.184.2.2 mask [1/2] . . . . .	363
7.184.2.3 version . . . . .	363
7.184.2.4 value [2/2] . . . . .	363
7.184.2.5 mask [2/2] . . . . .	364
7.184.2.6 channel_type . . . . .	364
7.184.2.7 proto_id . . . . .	364

7.184.2.8 comm_opt . . . . .	364
7.185vtss_phy_ts_alt_clock_mode_s Struct Reference . . . . .	364
7.185.1 Detailed Description . . . . .	365
7.185.2 Field Documentation . . . . .	365
7.185.2.1 pps_ls_lpbk . . . . .	365
7.185.2.2 ls_lpbk . . . . .	365
7.185.2.3 ls_pps_lpbk . . . . .	365
7.186vtss_phy_ts_eng_init_conf_t Struct Reference . . . . .	365
7.186.1 Detailed Description . . . . .	366
7.186.2 Field Documentation . . . . .	366
7.186.2.1 eng_used . . . . .	366
7.186.2.2 encaps_type . . . . .	366
7.186.2.3 flow_match_mode . . . . .	366
7.186.2.4 flow_st_index . . . . .	367
7.186.2.5 flow_end_index . . . . .	367
7.187vtss_phy_ts_engine_action_t Struct Reference . . . . .	367
7.187.1 Detailed Description . . . . .	367
7.187.2 Field Documentation . . . . .	368
7.187.2.1 action_ptp . . . . .	368
7.187.2.2 action_gen . . . . .	368
7.187.2.3 ptp_conf . . . . .	368
7.187.2.4 oam_conf . . . . .	368
7.187.2.5 gen_conf . . . . .	368
7.187.2.6 action . . . . .	369
7.188vtss_phy_ts_engine_flow_conf_t Struct Reference . . . . .	369
7.188.1 Detailed Description . . . . .	369
7.188.2 Field Documentation . . . . .	369
7.188.2.1 eng_mode . . . . .	369
7.188.2.2 channel_map . . . . .	370
7.188.2.3 ptp . . . . .	370

---

7.188.2.4 oam . . . . .	370
7.188.2.5 gen . . . . .	370
7.188.2.6 flow_conf . . . . .	370
7.189vtss_phy_ts_eth_conf_t Struct Reference . . . . .	371
7.189.1 Detailed Description . . . . .	371
7.189.2 Field Documentation . . . . .	372
7.189.2.1 pbb_en . . . . .	372
7.189.2.2 etype . . . . .	372
7.189.2.3 tpid . . . . .	372
7.189.2.4 comm_opt . . . . .	372
7.189.2.5 flow_en . . . . .	373
7.189.2.6 addr_match_mode . . . . .	373
7.189.2.7 addr_match_select . . . . .	373
7.189.2.8 mac_addr . . . . .	373
7.189.2.9 vlan_check . . . . .	373
7.189.2.10num_tag . . . . .	374
7.189.2.11outer_tag_type . . . . .	374
7.189.2.12inner_tag_type . . . . .	374
7.189.2.13tag_range_mode . . . . .	374
7.189.2.14upper . . . . .	374
7.189.2.15lower . . . . .	375
7.189.2.16range [1/2] . . . . .	375
7.189.2.17val [1/2] . . . . .	375
7.189.2.18mask [1/2] . . . . .	375
7.189.2.19value [1/2] . . . . .	375
7.189.2.20outer_tag . . . . .	375
7.189.2.21range [2/2] . . . . .	376
7.189.2.22value [2/2] . . . . .	376
7.189.2.23val [2/2] . . . . .	376
7.189.2.24mask [2/2] . . . . .	376

---

7.189.2.25_tag . . . . .	376
7.189.2.26_inner_tag . . . . .	376
7.189.2.27_low_opt . . . . .	377
7.190vtss_phy_ts_fifo_conf_t Struct Reference . . . . .	377
7.190.1 Detailed Description . . . . .	377
7.190.2 Field Documentation . . . . .	377
7.190.2.1 detect_only . . . . .	377
7.190.2.2 eng_recov . . . . .	378
7.190.2.3 eng_minE . . . . .	378
7.190.2.4 skip_rev_check . . . . .	378
7.191vtss_phy_ts_fifo_sig_t Struct Reference . . . . .	378
7.191.1 Detailed Description . . . . .	379
7.191.2 Field Documentation . . . . .	379
7.191.2.1 sig_mask . . . . .	379
7.191.2.2 msg_type . . . . .	379
7.191.2.3 domain_num . . . . .	379
7.191.2.4 src_port_identity . . . . .	379
7.191.2.5 sequence_id . . . . .	380
7.191.2.6 dest_ip . . . . .	380
7.191.2.7 src_ip . . . . .	380
7.191.2.8 dest_mac . . . . .	380
7.192vtss_phy_ts_gen_conf_t Struct Reference . . . . .	380
7.192.1 Detailed Description . . . . .	381
7.192.2 Field Documentation . . . . .	381
7.192.2.1 flow_offset . . . . .	381
7.192.2.2 next_prot_offset . . . . .	381
7.192.2.3 comm_opt . . . . .	381
7.192.2.4 flow_en . . . . .	382
7.192.2.5 data . . . . .	382
7.192.2.6 mask . . . . .	382

---

7.192.2.7 <code>flow_opt</code>	382
7.193 <code>vtss_phy_ts_generic_action_t</code> Struct Reference	382
7.193.1 Detailed Description	383
7.193.2 Field Documentation	383
7.193.2.1 <code>enable</code>	383
7.193.2.2 <code>channel_map</code>	383
7.193.2.3 <code>flow_id</code>	383
7.193.2.4 <code>data</code>	384
7.193.2.5 <code>mask</code>	384
7.193.2.6 <code>ts_type</code>	384
7.193.2.7 <code>ts_offset</code>	384
7.194 <code>vtss_phy_ts_generic_flow_conf_t</code> Struct Reference	384
7.194.1 Detailed Description	385
7.194.2 Field Documentation	385
7.194.2.1 <code>eth1_opt</code>	385
7.194.2.2 <code>gen_opt</code>	385
7.195 <code>vtss_phy_ts_ietf_mpls_ach_oam_conf_t</code> Struct Reference	385
7.195.1 Detailed Description	386
7.195.2 Field Documentation	386
7.195.2.1 <code>delaym_type</code>	386
7.195.2.2 <code>ts_format</code>	386
7.195.2.3 <code>ds</code>	386
7.196 <code>vtss_phy_ts_init_conf_t</code> Struct Reference	386
7.196.1 Detailed Description	387
7.196.2 Field Documentation	387
7.196.2.1 <code>clk_freq</code>	387
7.196.2.2 <code>clk_src</code>	387
7.196.2.3 <code>rx_ts_pos</code>	388
7.196.2.4 <code>rx_ts_len</code>	388
7.196.2.5 <code>tx_fifo_mode</code>	388

7.196.2.6 tx_ts_len . . . . .	388
7.196.2.7 tx_fifo_spi_conf . . . . .	388
7.196.2.8 tx_fifo_hi_clk_cyos . . . . .	389
7.196.2.9 tx_fifo_lo_clk_cyos . . . . .	389
7.196.2.10xaui_sel_8487 . . . . .	389
7.196.2.11tc_op_mode . . . . .	389
7.196.2.12auto_clear_ls . . . . .	389
7.196.2.13macsec_ena . . . . .	390
7.196.2.14chk_ing_modified . . . . .	390
7.196.2.15one_step_txfifo . . . . .	390
<b>7.197vtss_phy_ts_ip_conf_t Struct Reference</b> . . . . .	390
<b>7.197.1 Detailed Description</b> . . . . .	391
<b>7.197.2 Field Documentation</b> . . . . .	391
7.197.2.1 ip_mode . . . . .	391
7.197.2.2 sport_val . . . . .	392
7.197.2.3 sport_mask . . . . .	392
7.197.2.4 dport_val . . . . .	392
7.197.2.5 dport_mask . . . . .	392
7.197.2.6 comm_opt . . . . .	392
7.197.2.7 flow_en . . . . .	393
7.197.2.8 match_mode . . . . .	393
7.197.2.9 addr . . . . .	393
7.197.2.10mask . . . . .	393
7.197.2.11ipv4 . . . . .	393
7.197.2.12pv6 . . . . .	394
7.197.2.13p_addr . . . . .	394
7.197.2.14flow_opt . . . . .	394
<b>7.198vtss_phy_ts_mpls_conf_t Struct Reference</b> . . . . .	394
<b>7.198.1 Detailed Description</b> . . . . .	395
<b>7.198.2 Field Documentation</b> . . . . .	395

7.198.2.1 cw_en . . . . .	395
7.198.2.2 comm_opt . . . . .	395
7.198.2.3 flow_en . . . . .	395
7.198.2.4 stack_depth . . . . .	395
7.198.2.5 stack_ref_point . . . . .	396
7.198.2.6 top . . . . .	396
7.198.2.7 frst_lvl_after_top . . . . .	396
7.198.2.8 snd_lvl_after_top . . . . .	396
7.198.2.9 thrd_lvl_after_top . . . . .	396
7.198.2.10 top_down . . . . .	397
7.198.2.11 end . . . . .	397
7.198.2.12 frst_lvl_before_end . . . . .	397
7.198.2.13 snd_lvl_before_end . . . . .	397
7.198.2.14 hrd_lvl_before_end . . . . .	397
7.198.2.15 bottom_up . . . . .	397
7.198.2.16 stack_level . . . . .	398
7.198.2.17 flow_opt . . . . .	398
7.199 vtss_phy_ts_mpls_lvl_rng_t Struct Reference . . . . .	398
7.199.1 Detailed Description . . . . .	398
7.199.2 Field Documentation . . . . .	398
7.199.2.1 lower . . . . .	398
7.199.2.2 upper . . . . .	399
7.200 vtss_phy_ts_nphase_status_t Struct Reference . . . . .	399
7.200.1 Detailed Description . . . . .	399
7.200.2 Field Documentation . . . . .	399
7.200.2.1 enable . . . . .	399
7.200.2.2 CALIB_ERR . . . . .	400
7.200.2.3 CALIB_DONE . . . . .	400
7.201 vtss_phy_ts_oam_engine_action_t Struct Reference . . . . .	400
7.201.1 Detailed Description . . . . .	400

---

7.201.2 Field Documentation . . . . .	401
7.201.2.1 enable . . . . .	401
7.201.2.2 y1731_en . . . . .	401
7.201.2.3 channel_map . . . . .	401
7.201.2.4 version . . . . .	401
7.201.2.5 y1731_oam_conf . . . . .	401
7.201.2.6 ietf_oam_conf . . . . .	402
7.201.2.7 oam_conf . . . . .	402
7.202vtss_phy_ts_oam_engine_flow_conf_t Struct Reference . . . . .	402
7.202.1 Detailed Description . . . . .	402
7.202.2 Field Documentation . . . . .	402
7.202.2.1 eth1_opt . . . . .	403
7.202.2.2 eth2_opt . . . . .	403
7.202.2.3 mpls_opt . . . . .	403
7.202.2.4 ach_opt . . . . .	403
7.203vtss_phy_ts_pps_config_s Struct Reference . . . . .	403
7.203.1 Detailed Description . . . . .	404
7.203.2 Field Documentation . . . . .	404
7.203.2.1 pps_width_adj . . . . .	404
7.203.2.2 pps_offset . . . . .	404
7.203.2.3 pps_output_enable . . . . .	404
7.204vtss_phy_ts_ptp_conf_t Struct Reference . . . . .	405
7.204.1 Detailed Description . . . . .	405
7.204.2 Field Documentation . . . . .	405
7.204.2.1 range_en . . . . .	405
7.204.2.2 val . . . . .	405
7.204.2.3 mask . . . . .	406
7.204.2.4 value . . . . .	406
7.204.2.5 upper . . . . .	406
7.204.2.6 lower . . . . .	406

7.204.2.7 range . . . . .	406
7.204.2.8 domain . . . . .	406
7.205vtss_phy_ts_ptp_engine_action_t Struct Reference . . . . .	407
7.205.1 Detailed Description . . . . .	407
7.205.2 Field Documentation . . . . .	407
7.205.2.1 enable . . . . .	407
7.205.2.2 channel_map . . . . .	407
7.205.2.3 ptp_conf . . . . .	408
7.205.2.4 clk_mode . . . . .	408
7.205.2.5 delaym_type . . . . .	408
7.205.2.6 cf_update . . . . .	408
7.206vtss_phy_ts_ptp_engine_flow_conf_t Struct Reference . . . . .	408
7.206.1 Detailed Description . . . . .	409
7.206.2 Field Documentation . . . . .	409
7.206.2.1 eth1_opt . . . . .	409
7.206.2.2 eth2_opt . . . . .	409
7.206.2.3 ip1_opt . . . . .	409
7.206.2.4 ip2_opt . . . . .	410
7.206.2.5 mpls_opt . . . . .	410
7.206.2.6 ach_opt . . . . .	410
7.207vtss_phy_ts_sertod_conf_t Struct Reference . . . . .	410
7.207.1 Detailed Description . . . . .	410
7.207.2 Field Documentation . . . . .	411
7.207.2.1 ip_enable . . . . .	411
7.207.2.2 op_enable . . . . .	411
7.207.2.3 ls_inv . . . . .	411
7.208vtss_phy_ts_stats_t Struct Reference . . . . .	411
7.208.1 Detailed Description . . . . .	412
7.208.2 Field Documentation . . . . .	412
7.208.2.1 ingr_pream_shrink_err . . . . .	412

7.208.2.2 egr_pream_shrink_err . . . . .	412
7.208.2.3 ingr_fcs_err . . . . .	412
7.208.2.4 egr_fcs_err . . . . .	413
7.208.2.5 ingr_frm_mod_cnt . . . . .	413
7.208.2.6 egr_frm_mod_cnt . . . . .	413
7.208.2.7 ts_fifo_tx_cnt . . . . .	413
7.208.2.8 ts_fifo_drop_cnt . . . . .	413
7.209vtss_phy_ts_y1731_oam_conf_t Struct Reference . . . . .	414
7.209.1 Detailed Description . . . . .	414
7.209.2 Field Documentation . . . . .	414
7.209.2.1 range_en . . . . .	414
7.209.2.2 delaym_type . . . . .	414
7.209.2.3 val . . . . .	415
7.209.2.4 mask . . . . .	415
7.209.2.5 value . . . . .	415
7.209.2.6 upper . . . . .	415
7.209.2.7 lower . . . . .	415
7.209.2.8 range . . . . .	415
7.209.2.9 meg_level . . . . .	416
7.210vtss_phy_type_t Struct Reference . . . . .	416
7.210.1 Detailed Description . . . . .	416
7.210.2 Field Documentation . . . . .	416
7.210.2.1 part_number . . . . .	416
7.210.2.2 revision . . . . .	417
7.210.2.3 port_cnt . . . . .	417
7.210.2.4 channel_id . . . . .	417
7.210.2.5 base_port_no . . . . .	417
7.210.2.6 phy_api_base_no . . . . .	417
7.211vtss_phy_veriphy_result_t Struct Reference . . . . .	418
7.211.1 Detailed Description . . . . .	418

<b>7.211.2 Field Documentation</b>	418
7.211.2.1 link	418
7.211.2.2 status	418
7.211.2.3 length	419
<b>7.212vtss_phy_wol_conf_t Struct Reference</b>	419
<b>7.212.1 Detailed Description</b>	419
<b>7.212.2 Field Documentation</b>	419
7.212.2.1 secure_on_enable	419
7.212.2.2 wol_mac	420
7.212.2.3 wol_pass	420
7.212.2.4 wol_passwd_len	420
7.212.2.5 magic_pkt_cnt	420
<b>7.213vtss_pi_conf_t Struct Reference</b>	420
<b>7.213.1 Detailed Description</b>	421
<b>7.213.2 Field Documentation</b>	421
7.213.2.1 cs_wait_ns	421
<b>7.214vtss_policer_ext_t Struct Reference</b>	421
<b>7.214.1 Detailed Description</b>	422
<b>7.214.2 Field Documentation</b>	422
7.214.2.1 frame_rate	422
7.214.2.2 dp_bypass_level	422
7.214.2.3 unicast	422
7.214.2.4 multicast	422
7.214.2.5 broadcast	423
7.214.2.6 uc_no_flood	423
7.214.2.7 mc_no_flood	423
7.214.2.8 flooded	423
7.214.2.9 learning	423
7.214.2.10to_cpu	424
7.214.2.11cpu_queue	424

7.214.2.12limit_noncpu_traffic . . . . .	424
7.214.2.13limit_cpu_traffic . . . . .	424
7.214.2.14flow_control . . . . .	424
7.215vtss_policer_t Struct Reference . . . . .	425
7.215.1 Detailed Description . . . . .	425
7.215.2 Field Documentation . . . . .	425
7.215.2.1 level . . . . .	425
7.215.2.2 rate . . . . .	425
7.216vtss_port_bridge_counters_t Struct Reference . . . . .	425
7.216.1 Detailed Description . . . . .	426
7.216.2 Field Documentation . . . . .	426
7.216.2.1 dot1dTpPortInDiscards . . . . .	426
7.217vtss_port_clause_37_adv_t Struct Reference . . . . .	426
7.217.1 Detailed Description . . . . .	426
7.217.2 Field Documentation . . . . .	427
7.217.2.1 fdx . . . . .	427
7.217.2.2 hdx . . . . .	427
7.217.2.3 symmetric_pause . . . . .	427
7.217.2.4 asymmetric_pause . . . . .	427
7.217.2.5 remote_fault . . . . .	427
7.217.2.6 acknowledge . . . . .	428
7.217.2.7 next_page . . . . .	428
7.218vtss_port_clause_37_control_t Struct Reference . . . . .	428
7.218.1 Detailed Description . . . . .	428
7.218.2 Field Documentation . . . . .	428
7.218.2.1 enable . . . . .	429
7.218.2.2 advertisement . . . . .	429
7.219vtss_port_conf_t Struct Reference . . . . .	429
7.219.1 Detailed Description . . . . .	430
7.219.2 Field Documentation . . . . .	430

7.219.2.1 if_type . . . . .	430
7.219.2.2 sd_enable . . . . .	430
7.219.2.3 sd_active_high . . . . .	430
7.219.2.4 sd_internal . . . . .	430
7.219.2.5 frame_gaps . . . . .	431
7.219.2.6 power_down . . . . .	431
7.219.2.7 speed . . . . .	431
7.219.2.8 fdx . . . . .	431
7.219.2.9 flow_control . . . . .	431
7.219.2.10max_frame_length . . . . .	432
7.219.2.11frame_length_chk . . . . .	432
7.219.2.12max_tags . . . . .	432
7.219.2.13exc_col_cont . . . . .	432
7.219.2.14xaui_rx_lane_flip . . . . .	432
7.219.2.15xaui_tx_lane_flip . . . . .	433
7.219.2.16oop . . . . .	433
7.219.2.17serdes . . . . .	433
7.220vtss_port_counters_t Struct Reference . . . . .	433
7.220.1 Detailed Description . . . . .	434
7.220.2 Field Documentation . . . . .	434
7.220.2.1 rmon . . . . .	434
7.220.2.2 if_group . . . . .	434
7.220.2.3 ethernet_like . . . . .	434
7.220.2.4 bridge . . . . .	434
7.220.2.5 prop . . . . .	435
7.221vtss_port_ethernet_like_counters_t Struct Reference . . . . .	435
7.221.1 Detailed Description . . . . .	435
7.221.2 Field Documentation . . . . .	435
7.221.2.1 dot3StatsAlignmentErrors . . . . .	436
7.221.2.2 dot3StatsFCSErrors . . . . .	436

---

7.221.2.3 dot3StatsFrameTooLongs . . . . .	436
7.221.2.4 dot3StatsSymbolErrors . . . . .	436
7.221.2.5 dot3ControlInUnknownOpcodes . . . . .	436
7.221.2.6 dot3InPauseFrames . . . . .	437
7.221.2.7 dot3StatsSingleCollisionFrames . . . . .	437
7.221.2.8 dot3StatsMultipleCollisionFrames . . . . .	437
7.221.2.9 dot3StatsDeferredTransmissions . . . . .	437
7.221.2.10 dot3StatsLateCollisions . . . . .	437
7.221.2.11 dot3StatsExcessiveCollisions . . . . .	438
7.221.2.12 dot3StatsCarrierSenseErrors . . . . .	438
7.221.2.13 dot3OutPauseFrames . . . . .	438
7.222 vtss_port_flow_control_conf_t Struct Reference . . . . .	438
7.222.1 Detailed Description . . . . .	439
7.222.2 Field Documentation . . . . .	439
7.222.2.1 obey . . . . .	439
7.222.2.2 generate . . . . .	439
7.222.2.3 smac . . . . .	439
7.222.2.4 pfc . . . . .	439
7.223 vtss_port_frame_gaps_t Struct Reference . . . . .	440
7.223.1 Detailed Description . . . . .	440
7.223.2 Field Documentation . . . . .	440
7.223.2.1 hdx_gap_1 . . . . .	440
7.223.2.2 hdx_gap_2 . . . . .	440
7.223.2.3 fdx_gap . . . . .	441
7.224 vtss_port_if_group_counters_t Struct Reference . . . . .	441
7.224.1 Detailed Description . . . . .	441
7.224.2 Field Documentation . . . . .	441
7.224.2.1 ifInOctets . . . . .	442
7.224.2.2 ifInUcastPkts . . . . .	442
7.224.2.3 ifInMulticastPkts . . . . .	442

7.224.2.4 ifInBroadcastPkts . . . . .	442
7.224.2.5 ifInNUcastPkts . . . . .	442
7.224.2.6 ifInDiscards . . . . .	443
7.224.2.7 ifInErrors . . . . .	443
7.224.2.8 ifOutOctets . . . . .	443
7.224.2.9 ifOutUcastPkts . . . . .	443
7.224.2.10 ifOutMulticastPkts . . . . .	443
7.224.2.11 ifOutBroadcastPkts . . . . .	444
7.224.2.12 ifOutNUcastPkts . . . . .	444
7.224.2.13 ifOutDiscards . . . . .	444
7.224.2.14 ifOutErrors . . . . .	444
7.225 vtss_port_ifh_t Struct Reference . . . . .	444
7.225.1 Detailed Description . . . . .	445
7.225.2 Field Documentation . . . . .	445
7.225.2.1 ena_ifh_header . . . . .	445
7.226 vtss_port_map_t Struct Reference . . . . .	445
7.226.1 Detailed Description . . . . .	445
7.226.2 Field Documentation . . . . .	446
7.226.2.1 chip_port . . . . .	446
7.226.2.2 chip_no . . . . .	446
7.226.2.3 miim_controller . . . . .	446
7.226.2.4 miim_addr . . . . .	446
7.226.2.5 miim_chip_no . . . . .	447
7.227 vtss_port_proprietary_counters_t Struct Reference . . . . .	447
7.227.1 Detailed Description . . . . .	447
7.227.2 Field Documentation . . . . .	447
7.227.2.1 rx_prio . . . . .	447
7.227.2.2 tx_prio . . . . .	448
7.228 vtss_port_rmon_counters_t Struct Reference . . . . .	448
7.228.1 Detailed Description . . . . .	449

---

7.228.2 Field Documentation . . . . .	449
7.228.2.1 rx_etherStatsDropEvents . . . . .	449
7.228.2.2 rx_etherStatsOctets . . . . .	449
7.228.2.3 rx_etherStatsPkts . . . . .	449
7.228.2.4 rx_etherStatsBroadcastPkts . . . . .	449
7.228.2.5 rx_etherStatsMulticastPkts . . . . .	450
7.228.2.6 rx_etherStatsCRCAccErrors . . . . .	450
7.228.2.7 rx_etherStatsUndersizePkts . . . . .	450
7.228.2.8 rx_etherStatsOversizePkts . . . . .	450
7.228.2.9 rx_etherStatsFragments . . . . .	450
7.228.2.10 rx_etherStatsJabbers . . . . .	451
7.228.2.11 rx_etherStatsPkts64Octets . . . . .	451
7.228.2.12 rx_etherStatsPkts65to127Octets . . . . .	451
7.228.2.13 rx_etherStatsPkts128to255Octets . . . . .	451
7.228.2.14 rx_etherStatsPkts256to511Octets . . . . .	451
7.228.2.15 rx_etherStatsPkts512to1023Octets . . . . .	452
7.228.2.16 rx_etherStatsPkts1024to1518Octets . . . . .	452
7.228.2.17 rx_etherStatsPkts1519toMaxOctets . . . . .	452
7.228.2.18 rx_etherStatsDropEvents . . . . .	452
7.228.2.19 rx_etherStatsOctets . . . . .	452
7.228.2.20 rx_etherStatsPkts . . . . .	453
7.228.2.21 tx_etherStatsBroadcastPkts . . . . .	453
7.228.2.22 tx_etherStatsMulticastPkts . . . . .	453
7.228.2.23 tx_etherStatsCollisions . . . . .	453
7.228.2.24 tx_etherStatsPkts64Octets . . . . .	453
7.228.2.25 tx_etherStatsPkts65to127Octets . . . . .	454
7.228.2.26 tx_etherStatsPkts128to255Octets . . . . .	454
7.228.2.27 tx_etherStatsPkts256to511Octets . . . . .	454
7.228.2.28 tx_etherStatsPkts512to1023Octets . . . . .	454
7.228.2.29 tx_etherStatsPkts1024to1518Octets . . . . .	454

7.228.2.30x_etherStatsPkts1519toMaxOctets . . . . .	455
7.229vtss_port_serdes_conf_t Struct Reference . . . . .	455
7.229.1 Detailed Description . . . . .	455
7.229.2 Field Documentation . . . . .	455
7.229.2.1 sfp_dac . . . . .	455
7.230vtss_port_sgmii_aneg_t Struct Reference . . . . .	456
7.230.1 Detailed Description . . . . .	456
7.230.2 Field Documentation . . . . .	456
7.230.2.1 link . . . . .	456
7.230.2.2 fdx . . . . .	456
7.230.2.3 hdx . . . . .	457
7.230.2.4 speed_10M . . . . .	457
7.230.2.5 speed_100M . . . . .	457
7.230.2.6 speed_1G . . . . .	457
7.230.2.7 aneg_complete . . . . .	457
7.231vtss_port_status_t Struct Reference . . . . .	458
7.231.1 Detailed Description . . . . .	458
7.231.2 Field Documentation . . . . .	458
7.231.2.1 link_down . . . . .	458
7.231.2.2 link . . . . .	458
7.231.2.3 speed . . . . .	459
7.231.2.4 fdx . . . . .	459
7.231.2.5 remote_fault . . . . .	459
7.231.2.6 aneg_complete . . . . .	459
7.231.2.7 unidirectional_ability . . . . .	459
7.231.2.8 aneg . . . . .	460
7.231.2.9 mdi_cross . . . . .	460
7.231.2.10fiber . . . . .	460
7.231.2.11copper . . . . .	460
7.232vtss_qce_action_t Struct Reference . . . . .	460

7.232.1 Detailed Description	461
7.232.2 Field Documentation	461
7.232.2.1 prio_enable	461
7.232.2.2 prio	461
7.232.2.3 dp_enable	461
7.232.2.4 dp	462
7.232.2.5 dscp_enable	462
7.232.2.6 dscp	462
7.233vtss_qce_frame_etype_t Struct Reference	462
7.233.1 Detailed Description	462
7.233.2 Field Documentation	463
7.233.2.1 etype	463
7.233.2.2 data	463
7.234vtss_qce_frame_ipv4_t Struct Reference	463
7.234.1 Detailed Description	463
7.234.2 Field Documentation	464
7.234.2.1 fragment	464
7.234.2.2 dscp	464
7.234.2.3 proto	464
7.234.2.4 sip	464
7.234.2.5 sport	464
7.234.2.6 dport	465
7.235vtss_qce_frame_ipv6_t Struct Reference	465
7.235.1 Detailed Description	465
7.235.2 Field Documentation	465
7.235.2.1 dscp	465
7.235.2.2 proto	466
7.235.2.3 sip	466
7.235.2.4 sport	466
7.235.2.5 dport	466

7.236vtss_qce_frame_llc_t Struct Reference . . . . .	466
7.236.1 Detailed Description . . . . .	467
7.236.2 Field Documentation . . . . .	467
7.236.2.1 data . . . . .	467
7.237vtss_qce_frame_snap_t Struct Reference . . . . .	467
7.237.1 Detailed Description . . . . .	467
7.237.2 Field Documentation . . . . .	467
7.237.2.1 data . . . . .	468
7.238vtss_qce_key_t Struct Reference . . . . .	468
7.238.1 Detailed Description . . . . .	468
7.238.2 Field Documentation . . . . .	468
7.238.2.1 port_list . . . . .	468
7.238.2.2 mac . . . . .	469
7.238.2.3 tag . . . . .	469
7.238.2.4 type . . . . .	469
7.238.2.5 etype . . . . .	469
7.238.2.6 llc . . . . .	469
7.238.2.7 snap . . . . .	470
7.238.2.8 ipv4 . . . . .	470
7.238.2.9 ipv6 . . . . .	470
7.238.2.10 frame . . . . .	470
7.239vtss_qce_mac_t Struct Reference . . . . .	470
7.239.1 Detailed Description . . . . .	471
7.239.2 Field Documentation . . . . .	471
7.239.2.1 dmac_mc . . . . .	471
7.239.2.2 dmac_bc . . . . .	471
7.239.2.3 smac . . . . .	471
7.240vtss_qce_t Struct Reference . . . . .	472
7.240.1 Detailed Description . . . . .	472
7.240.2 Field Documentation . . . . .	472

---

7.240.2.1 id . . . . .	472
7.240.2.2 key . . . . .	472
7.240.2.3 action . . . . .	473
7.241vtss_qce_tag_t Struct Reference . . . . .	473
7.241.1 Detailed Description . . . . .	473
7.241.2 Field Documentation . . . . .	473
7.241.2.1 vid . . . . .	473
7.241.2.2 pcp . . . . .	474
7.241.2.3 dei . . . . .	474
7.241.2.4 tagged . . . . .	474
7.242vtss_qos_conf_t Struct Reference . . . . .	474
7.242.1 Detailed Description . . . . .	475
7.242.2 Field Documentation . . . . .	475
7.242.2.1 prios . . . . .	475
7.242.2.2 dscp_trust . . . . .	475
7.242.2.3 dscp_qos_class_map . . . . .	475
7.242.2.4 dscp_dp_level_map . . . . .	475
7.242.2.5 dscp_qos_map . . . . .	476
7.242.2.6 dscp_remark . . . . .	476
7.242.2.7 dscp_translate_map . . . . .	476
7.242.2.8 dscp_remap . . . . .	476
7.243vtss_qos_port_conf_t Struct Reference . . . . .	476
7.243.1 Detailed Description . . . . .	477
7.243.2 Field Documentation . . . . .	477
7.243.2.1 red . . . . .	477
7.243.2.2 policer_port . . . . .	478
7.243.2.3 policer_ext_port . . . . .	478
7.243.2.4 policer_queue . . . . .	478
7.243.2.5 shaper_port . . . . .	478
7.243.2.6 shaper_queue . . . . .	478

7.243.2.7 excess_enable . . . . .	479
7.243.2.8 default_prio . . . . .	479
7.243.2.9 usr_prio . . . . .	479
7.243.2.10 default_dpl . . . . .	479
7.243.2.11 default_dei . . . . .	479
7.243.2.12 ag_class_enable . . . . .	480
7.243.2.13 qos_class_map . . . . .	480
7.243.2.14 dp_level_map . . . . .	480
7.243.2.15 dscp_class_enable . . . . .	480
7.243.2.16 dscp_mode . . . . .	480
7.243.2.17 dscp_emode . . . . .	481
7.243.2.18 dscp_translate . . . . .	481
7.243.2.19 tag_remark_mode . . . . .	481
7.243.2.20 tag_default_pcp . . . . .	481
7.243.2.21 tag_default_dei . . . . .	481
7.243.2.22 tag_pcp_map . . . . .	482
7.243.2.23 tag_dei_map . . . . .	482
7.243.2.24 dwrr_enable . . . . .	482
7.243.2.25 queue_pct . . . . .	482
7.244 vtss_qs_conf_t Struct Reference . . . . .	482
7.244.1 Detailed Description . . . . .	483
7.244.2 Field Documentation . . . . .	483
7.244.2.1 mode . . . . .	483
7.244.2.2 oversubscription . . . . .	483
7.244.2.3 port_min . . . . .	483
7.244.2.4 port_max . . . . .	484
7.244.2.5 queue_min . . . . .	484
7.244.2.6 queue_max . . . . .	484
7.244.2.7 port . . . . .	484
7.245 vtss_rcpll_status_t Struct Reference . . . . .	484

---

7.245.1 Detailed Description . . . . .	485
7.245.2 Field Documentation . . . . .	485
7.245.2.1 out_of_range . . . . .	485
7.245.2.2 cal_error . . . . .	485
7.245.2.3 cal_not_done . . . . .	485
7.246vtss_red_t Struct Reference . . . . .	486
7.246.1 Detailed Description . . . . .	486
7.246.2 Field Documentation . . . . .	486
7.246.2.1 enable . . . . .	486
7.246.2.2 max_th . . . . .	486
7.246.2.3 min_th . . . . .	487
7.246.2.4 max_prob_1 . . . . .	487
7.246.2.5 max_prob_2 . . . . .	487
7.246.2.6 max_prob_3 . . . . .	487
7.247vtss_restart_status_t Struct Reference . . . . .	487
7.247.1 Detailed Description . . . . .	488
7.247.2 Field Documentation . . . . .	488
7.247.2.1 restart . . . . .	488
7.247.2.2 prev_version . . . . .	488
7.247.2.3 cur_version . . . . .	488
7.248vtss_routing_entry_t Struct Reference . . . . .	489
7.248.1 Detailed Description . . . . .	489
7.248.2 Field Documentation . . . . .	489
7.248.2.1 type . . . . .	489
7.248.2.2 ipv4_uc . . . . .	489
7.248.2.3 ipv6_uc . . . . .	490
7.248.2.4 route . . . . .	490
7.248.2.5 vlan . . . . .	490
7.249vtss_secure_on_passwd_t Struct Reference . . . . .	490
7.249.1 Detailed Description . . . . .	490

7.249.2 Field Documentation . . . . .	491
7.249.2.1 passwd . . . . .	491
7.250vtss_serd़es_macro_conf_t Struct Reference . . . . .	491
7.250.1 Detailed Description . . . . .	491
7.250.2 Field Documentation . . . . .	491
7.250.2.1 serdes1g_vdd . . . . .	491
7.250.2.2 serdes6g_vdd . . . . .	492
7.250.2.3 ib_cterm_ena . . . . .	492
7.250.2.4 qsgmii . . . . .	492
7.251vtss_sflow_port_conf_t Struct Reference . . . . .	492
7.251.1 Detailed Description . . . . .	493
7.251.2 Field Documentation . . . . .	493
7.251.2.1 type . . . . .	493
7.251.2.2 sampling_rate . . . . .	493
7.252vtss_sgpi़o_conf_t Struct Reference . . . . .	493
7.252.1 Detailed Description . . . . .	494
7.252.2 Field Documentation . . . . .	494
7.252.2.1 bmode . . . . .	494
7.252.2.2 bit_count . . . . .	494
7.252.2.3 port_conf . . . . .	494
7.253vtss_sgpi़o_port_conf_t Struct Reference . . . . .	494
7.253.1 Detailed Description . . . . .	495
7.253.2 Field Documentation . . . . .	495
7.253.2.1 enabled . . . . .	495
7.253.2.2 mode . . . . .	495
7.253.2.3 int_pol_high . . . . .	495
7.254vtss_sgpi़o_port_data_t Struct Reference . . . . .	496
7.254.1 Detailed Description . . . . .	496
7.254.2 Field Documentation . . . . .	496
7.254.2.1 value . . . . .	496

7.255vtss_shaper_t Struct Reference . . . . .	496
7.255.1 Detailed Description . . . . .	497
7.255.2 Field Documentation . . . . .	497
7.255.2.1 level . . . . .	497
7.255.2.2 rate . . . . .	497
7.256vtss_sublayer_status_t Struct Reference . . . . .	497
7.256.1 Detailed Description . . . . .	498
7.256.2 Field Documentation . . . . .	498
7.256.2.1 rx_link . . . . .	498
7.256.2.2 link_down . . . . .	498
7.256.2.3 rx_fault . . . . .	498
7.256.2.4 tx_fault . . . . .	498
7.257vtss_sync_clock_in_t Struct Reference . . . . .	499
7.257.1 Detailed Description . . . . .	499
7.257.2 Field Documentation . . . . .	499
7.257.2.1 port_no . . . . .	499
7.257.2.2 squelsh . . . . .	499
7.257.2.3 enable . . . . .	500
7.258vtss_sync_clock_out_t Struct Reference . . . . .	500
7.258.1 Detailed Description . . . . .	500
7.258.2 Field Documentation . . . . .	500
7.258.2.1 divider . . . . .	500
7.258.2.2 enable . . . . .	501
7.259vtss_tci_t Struct Reference . . . . .	501
7.259.1 Detailed Description . . . . .	501
7.259.2 Field Documentation . . . . .	501
7.259.2.1 vid . . . . .	501
7.259.2.2 cfi . . . . .	502
7.259.2.3 tagprior . . . . .	502
7.260vtss_timeofday_t Struct Reference . . . . .	502

7.260.1 Detailed Description . . . . .	502
7.260.2 Field Documentation . . . . .	502
7.260.2.1 sec [1/2] . . . . .	503
7.260.2.2 sec [2/2] . . . . .	503
7.261vtss_timestamp_t Struct Reference . . . . .	503
7.261.1 Detailed Description . . . . .	503
7.261.2 Field Documentation . . . . .	503
7.261.2.1 sec_msb . . . . .	504
7.261.2.2 seconds . . . . .	504
7.261.2.3 nanoseconds . . . . .	504
7.262vtss_trace_conf_t Struct Reference . . . . .	504
7.262.1 Detailed Description . . . . .	504
7.262.2 Field Documentation . . . . .	505
7.262.2.1 level . . . . .	505
7.263vtss_ts_ext_clock_mode_t Struct Reference . . . . .	505
7.263.1 Detailed Description . . . . .	505
7.263.2 Field Documentation . . . . .	505
7.263.2.1 one_pps_mode . . . . .	505
7.263.2.2 enable . . . . .	506
7.263.2.3 freq . . . . .	506
7.264vtss_ts_id_t Struct Reference . . . . .	506
7.264.1 Detailed Description . . . . .	506
7.264.2 Field Documentation . . . . .	506
7.264.2.1 ts_id . . . . .	507
7.265vtss_ts_internal_mode_t Struct Reference . . . . .	507
7.265.1 Detailed Description . . . . .	507
7.265.2 Field Documentation . . . . .	507
7.265.2.1 int_fmt . . . . .	507
7.266vtss_ts_operation_mode_t Struct Reference . . . . .	508
7.266.1 Detailed Description . . . . .	508

---

7.266.2 Field Documentation . . . . .	508
7.266.2.1 mode . . . . .	508
7.267vtss_ts_timestamp_alloc_t Struct Reference . . . . .	508
7.267.1 Detailed Description . . . . .	509
7.267.2 Field Documentation . . . . .	509
7.267.2.1 port_mask . . . . .	509
7.267.2.2 context . . . . .	509
7.267.2.3 cb . . . . .	509
7.268vtss_ts_timestamp_t Struct Reference . . . . .	509
7.268.1 Detailed Description . . . . .	510
7.268.2 Field Documentation . . . . .	510
7.268.2.1 ts . . . . .	510
7.268.2.2 id . . . . .	510
7.268.2.3 context . . . . .	510
7.268.2.4 ts_valid . . . . .	511
7.269vtss_vcap_ip_t Struct Reference . . . . .	511
7.269.1 Detailed Description . . . . .	511
7.269.2 Field Documentation . . . . .	511
7.269.2.1 value . . . . .	511
7.269.2.2 mask . . . . .	512
7.270vtss_vcap_u128_t Struct Reference . . . . .	512
7.270.1 Detailed Description . . . . .	512
7.270.2 Field Documentation . . . . .	512
7.270.2.1 value . . . . .	512
7.270.2.2 mask . . . . .	513
7.271vtss_vcap_u16_t Struct Reference . . . . .	513
7.271.1 Detailed Description . . . . .	513
7.271.2 Field Documentation . . . . .	513
7.271.2.1 value . . . . .	513
7.271.2.2 mask . . . . .	514

7.272vtss_vcap_u24_t Struct Reference . . . . .	514
7.272.1 Detailed Description . . . . .	514
7.272.2 Field Documentation . . . . .	514
7.272.2.1 value . . . . .	514
7.272.2.2 mask . . . . .	515
7.273vtss_vcap_u32_t Struct Reference . . . . .	515
7.273.1 Detailed Description . . . . .	515
7.273.2 Field Documentation . . . . .	515
7.273.2.1 value . . . . .	515
7.273.2.2 mask . . . . .	516
7.274vtss_vcap_u40_t Struct Reference . . . . .	516
7.274.1 Detailed Description . . . . .	516
7.274.2 Field Documentation . . . . .	516
7.274.2.1 value . . . . .	516
7.274.2.2 mask . . . . .	517
7.275vtss_vcap_u48_t Struct Reference . . . . .	517
7.275.1 Detailed Description . . . . .	517
7.275.2 Field Documentation . . . . .	517
7.275.2.1 value . . . . .	517
7.275.2.2 mask . . . . .	518
7.276vtss_vcap_u8_t Struct Reference . . . . .	518
7.276.1 Detailed Description . . . . .	518
7.276.2 Field Documentation . . . . .	518
7.276.2.1 value . . . . .	518
7.276.2.2 mask . . . . .	519
7.277vtss_vcap_udp_tcp_t Struct Reference . . . . .	519
7.277.1 Detailed Description . . . . .	519
7.277.2 Field Documentation . . . . .	519
7.277.2.1 in_range . . . . .	519
7.277.2.2 low . . . . .	520

---

<b>7.277.2.3 high</b>	520
<b>7.278vtss_vcap_vid_t Struct Reference</b>	520
<b>7.278.1 Detailed Description</b>	520
<b>7.278.2 Field Documentation</b>	520
<b>7.278.2.1 value</b>	521
<b>7.278.2.2 mask</b>	521
<b>7.279vtss_vcap_vr_t Struct Reference</b>	521
<b>7.279.1 Detailed Description</b>	521
<b>7.279.2 Field Documentation</b>	522
<b>7.279.2.1 type</b>	522
<b>7.279.2.2 value</b>	522
<b>7.279.2.3 mask</b>	522
<b>7.279.2.4 v</b>	522
<b>7.279.2.5 low</b>	522
<b>7.279.2.6 high</b>	523
<b>7.279.2.7 r</b>	523
<b>7.279.2.8 vr</b>	523
<b>7.280vtss_vce_action_t Struct Reference</b>	523
<b>7.280.1 Detailed Description</b>	523
<b>7.280.2 Field Documentation</b>	523
<b>7.280.2.1 vid</b>	524
<b>7.280.2.2 policy_no</b>	524
<b>7.281vtss_vce_frame_etype_t Struct Reference</b>	524
<b>7.281.1 Detailed Description</b>	524
<b>7.281.2 Field Documentation</b>	524
<b>7.281.2.1 etype</b>	525
<b>7.281.2.2 data</b>	525
<b>7.282vtss_vce_frame_ipv4_t Struct Reference</b>	525
<b>7.282.1 Detailed Description</b>	525
<b>7.282.2 Field Documentation</b>	525

7.282.2.1 fragment . . . . .	526
7.282.2.2 options . . . . .	526
7.282.2.3 dscp . . . . .	526
7.282.2.4 proto . . . . .	526
7.282.2.5 sip . . . . .	526
7.282.2.6 dport . . . . .	527
7.283vtss_vce_frame_ipv6_t Struct Reference . . . . .	527
7.283.1 Detailed Description . . . . .	527
7.283.2 Field Documentation . . . . .	527
7.283.2.1 dscp . . . . .	527
7.283.2.2 proto . . . . .	528
7.283.2.3 sip . . . . .	528
7.283.2.4 dport . . . . .	528
7.284vtss_vce_frame_llc_t Struct Reference . . . . .	528
7.284.1 Detailed Description . . . . .	528
7.284.2 Field Documentation . . . . .	529
7.284.2.1 data . . . . .	529
7.285vtss_vce_frame_snap_t Struct Reference . . . . .	529
7.285.1 Detailed Description . . . . .	529
7.285.2 Field Documentation . . . . .	529
7.285.2.1 data . . . . .	529
7.286vtss_vce_key_t Struct Reference . . . . .	530
7.286.1 Detailed Description . . . . .	530
7.286.2 Field Documentation . . . . .	530
7.286.2.1 port_list . . . . .	530
7.286.2.2 mac . . . . .	530
7.286.2.3 tag . . . . .	531
7.286.2.4 type . . . . .	531
7.286.2.5 etype . . . . .	531
7.286.2.6 llc . . . . .	531

7.286.2.7 snap . . . . .	531
7.286.2.8 ipv4 . . . . .	532
7.286.2.9 ipv6 . . . . .	532
7.286.2.10 frame . . . . .	532
7.287 vtss_vce_mac_t Struct Reference . . . . .	532
7.287.1 Detailed Description . . . . .	532
7.287.2 Field Documentation . . . . .	533
7.287.2.1 dmac_mc . . . . .	533
7.287.2.2 dmac_bc . . . . .	533
7.287.2.3 smac . . . . .	533
7.288 vtss_vce_t Struct Reference . . . . .	533
7.288.1 Detailed Description . . . . .	534
7.288.2 Field Documentation . . . . .	534
7.288.2.1 id . . . . .	534
7.288.2.2 key . . . . .	534
7.288.2.3 action . . . . .	534
7.289 vtss_vce_tag_t Struct Reference . . . . .	534
7.289.1 Detailed Description . . . . .	535
7.289.2 Field Documentation . . . . .	535
7.289.2.1 vid . . . . .	535
7.289.2.2 pcp . . . . .	535
7.289.2.3 dei . . . . .	535
7.289.2.4 tagged . . . . .	536
7.289.2.5 s_tag . . . . .	536
7.290 vtss_vcl_port_conf_t Struct Reference . . . . .	536
7.290.1 Detailed Description . . . . .	536
7.290.2 Field Documentation . . . . .	536
7.290.2.1 dmac_dip . . . . .	537
7.291 vtss_vid_mac_t Struct Reference . . . . .	537
7.291.1 Detailed Description . . . . .	537

7.291.2 Field Documentation . . . . .	537
7.291.2.1 vid . . . . .	537
7.291.2.2 mac . . . . .	538
7.292vtss_vlan_conf_t Struct Reference . . . . .	538
7.292.1 Detailed Description . . . . .	538
7.292.2 Field Documentation . . . . .	538
7.292.2.1 s_etype . . . . .	538
7.293vtss_vlan_port_conf_t Struct Reference . . . . .	539
7.293.1 Detailed Description . . . . .	539
7.293.2 Field Documentation . . . . .	539
7.293.2.1 port_type . . . . .	539
7.293.2.2 pvid . . . . .	539
7.293.2.3 untagged_vid . . . . .	540
7.293.2.4 frame_type . . . . .	540
7.293.2.5 ingress_filter . . . . .	540
7.294vtss_vlan_tag_t Struct Reference . . . . .	540
7.294.1 Detailed Description . . . . .	540
7.294.2 Field Documentation . . . . .	541
7.294.2.1 tpid . . . . .	541
7.294.2.2 pcp . . . . .	541
7.294.2.3 dei . . . . .	541
7.294.2.4 vid . . . . .	541
7.295vtss_vlan_trans_grp2vlan_conf_t Struct Reference . . . . .	542
7.295.1 Detailed Description . . . . .	542
7.295.2 Field Documentation . . . . .	542
7.295.2.1 group_id . . . . .	542
7.295.2.2 vid . . . . .	542
7.295.2.3 trans_vid . . . . .	543
7.296vtss_vlan_trans_port2grp_conf_t Struct Reference . . . . .	543
7.296.1 Detailed Description . . . . .	543

---

7.296.2 Field Documentation . . . . .	543
7.296.2.1 group_id . . . . .	543
7.296.2.2 ports . . . . .	544
7.297vtss_vlan_vid_conf_t Struct Reference . . . . .	544
7.297.1 Detailed Description . . . . .	544
7.297.2 Field Documentation . . . . .	544
7.297.2.1 learning . . . . .	544
7.297.2.2 mirror . . . . .	545
7.297.2.3 fid . . . . .	545
7.298vtss_vstax_conf_t Struct Reference . . . . .	545
7.298.1 Detailed Description . . . . .	545
7.298.2 Field Documentation . . . . .	545
7.298.2.1 upsid_0 . . . . .	546
7.298.2.2 upsid_1 . . . . .	546
7.298.2.3 port_0 . . . . .	546
7.298.2.4 port_1 . . . . .	546
7.298.2.5 cmeft_disable . . . . .	546
7.299vtss_vstax_glag_entry_t Struct Reference . . . . .	547
7.299.1 Detailed Description . . . . .	547
7.299.2 Field Documentation . . . . .	547
7.299.2.1 upsid . . . . .	547
7.299.2.2 upspn . . . . .	547
7.300vtss_vstax_port_conf_t Struct Reference . . . . .	547
7.300.1 Detailed Description . . . . .	548
7.300.2 Field Documentation . . . . .	548
7.300.2.1 ttl . . . . .	548
7.300.2.2 mirror . . . . .	548
7.301vtss_vstax_route_entry_t Struct Reference . . . . .	548
7.301.1 Detailed Description . . . . .	549
7.301.2 Field Documentation . . . . .	549

7.301.2.1 stack_port_a . . . . .	549
7.301.2.2 stack_port_b . . . . .	549
7.302vtss_vstax_route_table_t Struct Reference . . . . .	549
7.302.1 Detailed Description . . . . .	550
7.302.2 Field Documentation . . . . .	550
7.302.2.1 topology_type . . . . .	550
7.302.2.2 table . . . . .	550
7.303vtss_vstax_rx_header_t Struct Reference . . . . .	550
7.303.1 Detailed Description . . . . .	551
7.303.2 Field Documentation . . . . .	551
7.303.2.1 valid . . . . .	551
7.303.2.2 sp . . . . .	551
7.303.2.3 upsid . . . . .	551
7.303.2.4 port_no . . . . .	551
7.303.2.5 glag_no . . . . .	552
7.303.2.6 isdx . . . . .	552
7.304vtss_vstax_tx_header_t Struct Reference . . . . .	552
7.304.1 Detailed Description . . . . .	552
7.304.2 Field Documentation . . . . .	553
7.304.2.1 fwd_mode . . . . .	553
7.304.2.2 ttl . . . . .	553
7.304.2.3 prio . . . . .	553
7.304.2.4 upsid . . . . .	553
7.304.2.5 tci . . . . .	553
7.304.2.6 port_no . . . . .	554
7.304.2.7 chip_port . . . . .	554
7.304.2.8 glag_no . . . . .	554
7.304.2.9 queue_no . . . . .	554
7.304.2.10keep_ttl . . . . .	554
7.304.2.11dp . . . . .	555
7.305vtss_wol_mac_addr_t Struct Reference . . . . .	555
7.305.1 Detailed Description . . . . .	555
7.305.2 Field Documentation . . . . .	555
7.305.2.1 addr . . . . .	555

---

<b>8 File Documentation</b>	<b>557</b>
8.1 <a href="#">vtss_api/include/vtss/api/I2_types.h File Reference</a>	557
8.1.1 <a href="#">Detailed Description</a>	557
8.1.2 <a href="#">Enumeration Type Documentation</a>	557
8.1.2.1 <a href="#">vtss_sflow_type_t</a>	557
8.2 <a href="#">vtss_api/include/vtss/api/options.h File Reference</a>	558
8.2.1 <a href="#">Detailed Description</a>	560
8.2.2 <a href="#">Macro Definition Documentation</a>	560
8.2.2.1 <a href="#">VTSS_ARCH_JAGUAR_1</a>	560
8.2.2.2 <a href="#">VTSS_ARCH_JAGUAR_1_CE_SWITCH</a>	560
8.2.2.3 <a href="#">VTSS_FEATURE_EVC</a>	560
8.2.2.4 <a href="#">VTSS_FEATURE_E_TREE</a>	560
8.2.2.5 <a href="#">VTSS_FEATURE_TIMESTAMP</a>	561
8.2.2.6 <a href="#">VTSS_FEATURE_PHY_TIMESTAMP</a>	561
8.2.2.7 <a href="#">VTSS_FEATURE_VSTAX</a>	561
8.2.2.8 <a href="#">VTSS_FEATURE_AGGR_GLAG</a>	561
8.2.2.9 <a href="#">VTSS_FEATURE_VSTAX_V2</a>	561
8.2.2.10 <a href="#">VTSS_PHY_TS_SPI_CLK_THRU_PPS0</a>	562
8.2.2.11 <a href="#">VTSS_FEATURE_FAN [1/2]</a>	562
8.2.2.12 <a href="#">VTSS_FEATURE_PVLAN</a>	562
8.2.2.13 <a href="#">VTSS_OPT_TS_SPI_FPGA</a>	562
8.2.2.14 <a href="#">VTSS_CHIP_10G_PHY</a>	562
8.2.2.15 <a href="#">VTSS_FEATURE_MISC</a>	563
8.2.2.16 <a href="#">VTSS_FEATURE_SERIAL_GPIO</a>	563
8.2.2.17 <a href="#">VTSS_FEATURE_PORT_CONTROL</a>	563
8.2.2.18 <a href="#">VTSS_FEATURE_PORT_IFH</a>	563
8.2.2.19 <a href="#">VTSS_FEATURE_CLAUSE_37</a>	563
8.2.2.20 <a href="#">VTSS_FEATURE_10G</a>	564
8.2.2.21 <a href="#">VTSS_FEATURE_NPI</a>	564
8.2.2.22 <a href="#">VTSS_FEATURE_EXC_COL_CONT</a>	564

8.2.2.23	VTSS_FEATURE_PORT_CNT_ETHER_LIKE . . . . .	564
8.2.2.24	VTSS_FEATURE_PORT_CNT_BRIDGE . . . . .	564
8.2.2.25	VTSS_FEATURE_QOS . . . . .	565
8.2.2.26	VTSS_FEATURE_QCL . . . . .	565
8.2.2.27	VTSS_FEATURE_QCL_V2 . . . . .	565
8.2.2.28	VTSS_FEATURE_QOS_PORT_POLICER_EXT . . . . .	565
8.2.2.29	VTSS_FEATURE_QOS_PORT_POLICER_EXT_FPS . . . . .	565
8.2.2.30	VTSS_FEATURE_QOS_PORT_POLICER_EXT_FC . . . . .	566
8.2.2.31	VTSS_FEATURE_QOS_PORT_POLICER_EXT_DPBL . . . . .	566
8.2.2.32	VTSS_FEATURE_QOS_PORT_POLICER_EXT_TTM . . . . .	566
8.2.2.33	VTSS_FEATURE_QOS_QUEUE_POLICER . . . . .	566
8.2.2.34	VTSS_FEATURE_QOS_QUEUE_TX . . . . .	566
8.2.2.35	VTSS_FEATURE_QOS_SCHEDULER_V2 . . . . .	567
8.2.2.36	VTSS_FEATURE_QOS_TAG_REMARK_V2 . . . . .	567
8.2.2.37	VTSS_FEATURE_QOS_CLASSIFICATION_V2 . . . . .	567
8.2.2.38	VTSS_FEATURE_QOS_EGRESS_QUEUE_SHAPERS . . . . .	567
8.2.2.39	VTSS_FEATURE_QOS_EGRESS_QUEUE_SHAPERS_EB . . . . .	567
8.2.2.40	VTSS_FEATURE_QOS_DSCP_REMARK . . . . .	568
8.2.2.41	VTSS_FEATURE_QOS_DSCP_REMARK_V2 . . . . .	568
8.2.2.42	VTSS_FEATURE_QOS_WRED . . . . .	568
8.2.2.43	VTSS_FEATURE_PACKET . . . . .	568
8.2.2.44	VTSS_FEATURE_PACKET_TX . . . . .	568
8.2.2.45	VTSS_FEATURE_PACKET_RX . . . . .	569
8.2.2.46	VTSS_FEATURE_PACKET_GROUPING . . . . .	569
8.2.2.47	VTSS_FEATURE_PACKET_PORT_REG . . . . .	569
8.2.2.48	VTSS_FEATURE_LAYER2 . . . . .	569
8.2.2.49	VTSS_FEATURE_VLAN_PORT_V2 . . . . .	569
8.2.2.50	VTSS_FEATURE_VLAN_TX_TAG . . . . .	570
8.2.2.51	VTSS_FEATURE_VLAN_SVL . . . . .	570
8.2.2.52	VTSS_FEATURE_MAC_AGE_AUTO . . . . .	570

8.2.2.53	VTSS_FEATURE_MAC_CPU_QUEUE . . . . .	570
8.2.2.54	VTSS_FEATURE_LAYER3 . . . . .	570
8.2.2.55	VTSS_FEATURE_PORT_MUX . . . . .	571
8.2.2.56	VTSS_FEATURE_VCAP [1/2] . . . . .	571
8.2.2.57	VTSS_FEATURE_ACL . . . . .	571
8.2.2.58	VTSS_FEATURE_ACL_V1 . . . . .	571
8.2.2.59	VTSS_FEATURE_VCL . . . . .	571
8.2.2.60	VTSS_FEATURE_SYNCE . . . . .	572
8.2.2.61	VTSS_FEATURE_FAN [2/2] . . . . .	572
8.2.2.62	VTSS_FEATURE_EEE . . . . .	572
8.2.2.63	VTSS_FEATURE_SERDES_MACRO_SETTINGS . . . . .	572
8.2.2.64	VTSS_FEATURE_LED_POW_REDUC . . . . .	572
8.2.2.65	VTSS_FEATURE_10GBASE_KR . . . . .	573
8.2.2.66	VTSS_FEATURE_IRQ_CONTROL . . . . .	573
8.2.2.67	VTSS_OPT_VCORE_III . . . . .	573
8.2.2.68	VTSS_FEATURE_FDMA . . . . .	573
8.2.2.69	VTSS_FEATURE_AFI_FDMA . . . . .	573
8.2.2.70	VTSS_FEATURE_VLAN_TRANSLATION . . . . .	574
8.2.2.71	VTSS_FEATURE_SFLOW . . . . .	574
8.2.2.72	VTSS_PHY_10G_FIFO_SYNC . . . . .	574
8.2.2.73	VIPER_B_FIFO_RESET . . . . .	574
8.2.2.74	VTSS_FEATURE_QOS_POLICER_DLDB . . . . .	574
8.2.2.75	VTSS_CHIP CU PHY . . . . .	575
8.2.2.76	VTSS_OPT_TRACE . . . . .	575
8.2.2.77	VTSS_OPT_FDMA_IRQ_CONTEXT . . . . .	575
8.2.2.78	VTSS_OPT_FDMA_DEBUG . . . . .	575
8.2.2.79	VTSS_OPT_VAUI_EQ_CTRL . . . . .	575
8.2.2.80	VTSS_PHY_OPT_VERIPHYS . . . . .	576
8.2.2.81	VTSS_FEATURE_WARM_START [1/2] . . . . .	576
8.2.2.82	VTSS_FEATURE_SYNCE_10G . . . . .	576

8.2.2.83	VTSS_FEATURE_EDC_FW_LOAD . . . . .	576
8.2.2.84	VTSS_FEATURE_WIS . . . . .	576
8.2.2.85	VTSS_FEATURE_WARM_START [2/2] . . . . .	577
8.2.2.86	VTSS_ARCH_MALIBU . . . . .	577
8.2.2.87	VTSS_ARCH_MALIBU_B . . . . .	577
8.2.2.88	VTSS_ARCH_VENICE_C . . . . .	577
8.2.2.89	VTSS_FEATURE_VCAP [2/2] . . . . .	577
8.3	vtss_api/include/vtss/api/phy.h File Reference . . . . .	578
8.3.1	Detailed Description . . . . .	578
8.3.2	Macro Definition Documentation . . . . .	578
8.3.2.1	VTSS_PHY_POWER_ACTIPHY_BIT . . . . .	578
8.3.2.2	VTSS_PHY_POWER_DYNAMIC_BIT . . . . .	579
8.3.3	Enumeration Type Documentation . . . . .	579
8.3.3.1	vtss_phy_power_mode_t . . . . .	579
8.3.3.2	vtss_phy_veriphy_status_t . . . . .	579
8.4	vtss_api/include/vtss/api/port.h File Reference . . . . .	580
8.4.1	Detailed Description . . . . .	582
8.4.2	Macro Definition Documentation . . . . .	582
8.4.2.1	PORT_CAP_NONE . . . . .	582
8.4.2.2	PORT_CAP_AUTONEG . . . . .	582
8.4.2.3	PORT_CAP_10M_HDX . . . . .	582
8.4.2.4	PORT_CAP_10M_FDX . . . . .	582
8.4.2.5	PORT_CAP_100M_HDX . . . . .	583
8.4.2.6	PORT_CAP_100M_FDX . . . . .	583
8.4.2.7	PORT_CAP_1G_FDX . . . . .	583
8.4.2.8	PORT_CAP_2_5G_FDX . . . . .	583
8.4.2.9	PORT_CAP_5G_FDX . . . . .	583
8.4.2.10	PORT_CAP_10G_FDX . . . . .	584
8.4.2.11	PORT_CAP_FLOW_CTRL . . . . .	584
8.4.2.12	PORT_CAP_COPPER . . . . .	584

8.4.2.13	PORT_CAP_FIBER	584
8.4.2.14	PORT_CAP_DUAL_COPPER	584
8.4.2.15	PORT_CAP_DUAL_FIBER	585
8.4.2.16	PORT_CAP_SD_ENABLE	585
8.4.2.17	PORT_CAP_SD_HIGH	585
8.4.2.18	PORT_CAP_SD_INTERNAL	585
8.4.2.19	PORT_CAP_DUAL_FIBER_100FX	585
8.4.2.20	PORT_CAP_XAUI_LANE_FLIP	586
8.4.2.21	PORT_CAP_VTSS_10G_PHY	586
8.4.2.22	PORT_CAP_SFP_DETECT	586
8.4.2.23	PORT_CAP_STACKING	586
8.4.2.24	PORT_CAP_DUAL_SFP_DETECT	586
8.4.2.25	PORT_CAP_SFP_ONLY	587
8.4.2.26	PORT_CAP_DUAL_COPPER_100FX	587
8.4.2.27	PORT_CAP_HDX	587
8.4.2.28	PORT_CAP_TRI_SPEED_FDX	587
8.4.2.29	PORT_CAP_TRI_SPEED	587
8.4.2.30	PORT_CAP_1G_PHY	588
8.4.2.31	PORT_CAP_TRI_SPEED_COPPER	588
8.4.2.32	PORT_CAP_TRI_SPEED_FIBER	588
8.4.2.33	PORT_CAP_TRI_SPEED_DUAL_COPPER	588
8.4.2.34	PORT_CAP_TRI_SPEED_DUAL_FIBER	588
8.4.2.35	PORT_CAP_TRI_SPEED_DUAL_FIBER_100FX	589
8.4.2.36	PORT_CAP_ANY_FIBER	589
8.4.2.37	PORT_CAP_SPEED_DUAL_ANY_FIBER_FIXED_SPEED	589
8.4.2.38	PORT_CAP_SPEED_DUAL_ANY_FIBER	589
8.4.2.39	PORT_CAP_TRI_SPEED_DUAL_ANY_FIBER	589
8.4.2.40	PORT_CAP_TRI_SPEED_DUAL_ANY_FIBER_FIXED_SFP_SPEED	590
8.4.2.41	PORT_CAP_DUAL_FIBER_1000X	590
8.4.2.42	PORT_CAP_SFP_1G	590

8.4.2.43	PORT_CAP_SFP_2_5G . . . . .	590
8.4.2.44	PORT_CAP_SFP_SD_HIGH . . . . .	590
8.4.2.45	PORT_CAP_2_5G_TRI_SPEED_FDX . . . . .	591
8.4.2.46	PORT_CAP_2_5G_TRI_SPEED . . . . .	591
8.4.2.47	PORT_CAP_2_5G_TRI_SPEED_COPPER . . . . .	591
8.4.3	Typedef Documentation . . . . .	591
8.4.3.1	port_cap_t . . . . .	591
8.4.4	Enumeration Type Documentation . . . . .	591
8.4.4.1	vtss_port_speed_t . . . . .	591
8.4.4.2	vtss_fiber_port_speed_t . . . . .	592
8.5	vtss_api/include/vtss/api/types.h File Reference . . . . .	592
8.5.1	Detailed Description . . . . .	599
8.5.2	Macro Definition Documentation . . . . .	600
8.5.2.1	PRIlu64 . . . . .	600
8.5.2.2	PRIli64 . . . . .	600
8.5.2.3	PRIx64 . . . . .	600
8.5.2.4	VTSS_BIT64 . . . . .	600
8.5.2.5	VTSS_BITMASK64 . . . . .	601
8.5.2.6	VTSS_EXTRACT_BITFIELD64 . . . . .	601
8.5.2.7	VTSS_ENCODE_BITFIELD64 . . . . .	601
8.5.2.8	VTSS_ENCODE_BITMASK64 . . . . .	601
8.5.2.9	TRUE . . . . .	602
8.5.2.10	FALSE . . . . .	602
8.5.2.11	VTSS_PACKET_RATE_DISABLED . . . . .	602
8.5.2.12	VTSS_PORT_COUNT [1/2] . . . . .	602
8.5.2.13	VTSS_PORT_COUNT [2/2] . . . . .	602
8.5.2.14	VTSS_PORTS . . . . .	603
8.5.2.15	VTSS_PORT_NO_NONE . . . . .	603
8.5.2.16	VTSS_PORT_NO_CPU . . . . .	603
8.5.2.17	VTSS_PORT_NO_START . . . . .	603

8.5.2.18	VTSS_PORT_NO_END . . . . .	603
8.5.2.19	VTSS_PORT_ARRAY_SIZE . . . . .	604
8.5.2.20	VTSS_PORT_IS_PORT . . . . .	604
8.5.2.21	VTSS_PRIOS . . . . .	604
8.5.2.22	VTSS_PRIO_NO_NONE . . . . .	604
8.5.2.23	VTSS_PRIO_START . . . . .	604
8.5.2.24	VTSS_PRIO_END . . . . .	605
8.5.2.25	VTSS_PRIO_ARRAY_SIZE . . . . .	605
8.5.2.26	VTSS_QUEUES . . . . .	605
8.5.2.27	VTSS_QUEUE_START . . . . .	605
8.5.2.28	VTSS_QUEUE_END . . . . .	605
8.5.2.29	VTSS_QUEUE_ARRAY_SIZE . . . . .	606
8.5.2.30	VTSS_PCPS . . . . .	606
8.5.2.31	VTSS_PCP_START . . . . .	606
8.5.2.32	VTSS_PCP_END . . . . .	606
8.5.2.33	VTSS_PCP_ARRAY_SIZE . . . . .	606
8.5.2.34	VTSS_DEIS . . . . .	607
8.5.2.35	VTSS_DEI_START . . . . .	607
8.5.2.36	VTSS_DEI_END . . . . .	607
8.5.2.37	VTSS_DEI_ARRAY_SIZE . . . . .	607
8.5.2.38	VTSS_DPLS [1/2] . . . . .	607
8.5.2.39	VTSS_DPLS [2/2] . . . . .	608
8.5.2.40	VTSS_DPL_START . . . . .	608
8.5.2.41	VTSS_DPL_END . . . . .	608
8.5.2.42	VTSS_DPL_ARRAY_SIZE . . . . .	608
8.5.2.43	VTSS_BITRATE_DISABLED . . . . .	608
8.5.2.44	VTSS_VID_NULL . . . . .	609
8.5.2.45	VTSS_VID_DEFAULT . . . . .	609
8.5.2.46	VTSS_VID_RESERVED . . . . .	609
8.5.2.47	VTSS_VIDS . . . . .	609

8.5.2.48	VTSS_VID_ALL . . . . .	609
8.5.2.49	VTSSETYPE_VTSS . . . . .	610
8.5.2.50	VTSS_MAC_ADDR_SZ_BYTES . . . . .	610
8.5.2.51	MAC_ADDR_BROADCAST . . . . .	610
8.5.2.52	VTSS_EVCS . . . . .	610
8.5.2.53	VTSS_ISDX_NONE . . . . .	610
8.5.2.54	VTSS_AGGRS . . . . .	611
8.5.2.55	VTSS_AGGR_NO_NONE . . . . .	611
8.5.2.56	VTSS_AGGR_NO_START . . . . .	611
8.5.2.57	VTSS_AGGR_NO_END . . . . .	611
8.5.2.58	VTSS_GLAGS . . . . .	611
8.5.2.59	VTSS_GLAG_NO_NONE . . . . .	612
8.5.2.60	VTSS_GLAG_NO_START . . . . .	612
8.5.2.61	VTSS_GLAG_NO_END . . . . .	612
8.5.2.62	VTSS_GLAG_PORTS . . . . .	612
8.5.2.63	VTSS_GLAG_PORT_START . . . . .	612
8.5.2.64	VTSS_GLAG_PORT_END . . . . .	613
8.5.2.65	VTSS_GLAG_PORT_ARRAY_SIZE . . . . .	613
8.5.2.66	VTSS_PACKET_RX_QUEUE_CNT . . . . .	613
8.5.2.67	VTSS_PACKET_RX_GRP_CNT . . . . .	613
8.5.2.68	VTSS_PACKET_TX_GRP_CNT . . . . .	613
8.5.2.69	VTSS_PACKET_RX_QUEUE_NONE . . . . .	614
8.5.2.70	VTSS_PACKET_RX_QUEUE_START . . . . .	614
8.5.2.71	VTSS_PACKET_RX_QUEUE_END . . . . .	614
8.5.2.72	VTSS_ACL_POLICERS . . . . .	614
8.5.2.73	VTSS_ACL_POLICER_NO_START . . . . .	614
8.5.2.74	VTSS_ACL_POLICER_NO_END . . . . .	615
8.5.2.75	VTSS_ACL_POLICY_NO_NONE . . . . .	615
8.5.2.76	VTSS_ACL_POLICY_NO_MIN . . . . .	615
8.5.2.77	VTSS_ACL_POLICY_NO_MAX . . . . .	615

---

8.5.2.78	VTSS_ACL_POLICIES	615
8.5.2.79	VTSS_ACL_POLICY_NO_START	616
8.5.2.80	VTSS_ACL_POLICY_NO_END	616
8.5.2.81	VTSS_HQOS_COUNT	616
8.5.2.82	VTSS_HQOS_ID_NONE	616
8.5.2.83	VTSS_ONE_MIA	616
8.5.2.84	VTSS_ONE_MILL	617
8.5.2.85	VTSS_MAX_TIMEINTERVAL	617
8.5.2.86	VTSS_INTERVAL_SEC	617
8.5.2.87	VTSS_INTERVAL_MS	617
8.5.2.88	VTSS_INTERVAL_US	617
8.5.2.89	VTSS_INTERVAL_NS	618
8.5.2.90	VTSS_INTERVAL_PS	618
8.5.2.91	VTSS_SEC_NS_INTERVAL	618
8.5.2.92	VTSS_CLOCK_IDENTITY_LENGTH	618
8.5.2.93	VTSS_SYNC_CLK_PORT_ARRAY_SIZE	618
8.5.3	Typedef Documentation	619
8.5.3.1	i8	619
8.5.3.2	i16	619
8.5.3.3	i32	619
8.5.3.4	i64	619
8.5.3.5	u8	620
8.5.3.6	u16	620
8.5.3.7	u32	620
8.5.3.8	u64	620
8.5.3.9	BOOL	620
8.5.3.10	uintptr_t	621
8.5.3.11	vtss_mac_addr_t	621
8.5.3.12	vtss_isdx_t	621
8.5.3.13	vtss_packet_rx_grp_t	621

8.5.3.14	<a href="#">vtss_packet_tx_grp_t</a>	621
8.5.4	<a href="#">Enumeration Type Documentation</a>	621
8.5.4.1	<a href="#">anonymous enum</a>	621
8.5.4.2	<a href="#">vtss_mem_flags_t</a>	624
8.5.4.3	<a href="#">vtss_port_interface_t</a>	624
8.5.4.4	<a href="#">vtss_serdes_mode_t</a>	625
8.5.4.5	<a href="#">vtss_storm_policer_mode_t</a>	626
8.5.4.6	<a href="#">vtss_policer_type_t</a>	626
8.5.4.7	<a href="#">vtss_vlan_frame_t</a>	626
8.5.4.8	<a href="#">vtss_packet_reg_type_t</a>	627
8.5.4.9	<a href="#">vtss_vdd_t</a>	627
8.5.4.10	<a href="#">vtss_ip_type_t</a>	627
8.5.4.11	<a href="#">vtss_vcap_bit_t</a>	628
8.5.4.12	<a href="#">vtss_vcap_vr_type_t</a>	628
8.5.4.13	<a href="#">vtss_vcap_key_type_t</a>	628
8.5.4.14	<a href="#">vtss_ece_dir_t</a>	629
8.5.4.15	<a href="#">vtss_ece_pop_tag_t</a>	629
8.5.4.16	<a href="#">vtss_ece_inner_tag_type_t</a>	629
8.5.4.17	<a href="#">vtss_hqos_sch_mode_t</a>	630
8.6	<a href="#">vtss_api/include/vtss_ae_api.h File Reference</a>	630
8.6.1	<a href="#">Detailed Description</a>	630
8.7	<a href="#">vtss_api/include/vtss_afi_api.h File Reference</a>	630
8.7.1	<a href="#">Detailed Description</a>	630
8.8	<a href="#">vtss_api/include/vtss_aneg_api.h File Reference</a>	631
8.8.1	<a href="#">Detailed Description</a>	631
8.9	<a href="#">vtss_api/include/vtss_api.h File Reference</a>	631
8.9.1	<a href="#">Detailed Description</a>	631
8.10	<a href="#">vtss_api/include/vtss_evc_api.h File Reference</a>	631
8.10.1	<a href="#">Detailed Description</a>	633
8.10.2	<a href="#">Macro Definition Documentation</a>	633

8.10.2.1	VTSS_EVC_POLICERS . . . . .	634
8.10.2.2	VTSS_EVC_POLICER_ID_DISCARD . . . . .	634
8.10.2.3	VTSS_EVC_POLICER_ID_NONE . . . . .	634
8.10.2.4	VTSS_EVC_POLICER_ID_EVC . . . . .	634
8.10.2.5	VTSS_EVC_ID_NONE . . . . .	634
8.10.2.6	VTSS_ECE_ID_LAST . . . . .	635
8.10.3	Enumeration Type Documentation . . . . .	635
8.10.3.1	vtss_ece_type_t . . . . .	635
8.10.3.2	vtss_ece_port_t . . . . .	635
8.10.4	Function Documentation . . . . .	635
8.10.4.1	vtss_evc_port_conf_get() . . . . .	636
8.10.4.2	vtss_evc_port_conf_set() . . . . .	636
8.10.4.3	vtss_evc_policer_conf_get() . . . . .	636
8.10.4.4	vtss_evc_policer_conf_set() . . . . .	637
8.10.4.5	vtss_evc_add() . . . . .	637
8.10.4.6	vtss_evc_del() . . . . .	638
8.10.4.7	vtss_evc_get() . . . . .	638
8.10.4.8	vtss_ece_init() . . . . .	639
8.10.4.9	vtss_ece_add() . . . . .	639
8.10.4.10	vtss_ece_del() . . . . .	639
8.10.4.11	vtss_evc_counters_get() . . . . .	640
8.10.4.12	vtss_evc_counters_clear() . . . . .	640
8.10.4.13	vtss_ece_counters_get() . . . . .	641
8.10.4.14	vtss_ece_counters_clear() . . . . .	641
8.11	vtss_api/include/vtss_fdma_api.h File Reference . . . . .	642
8.11.1	Detailed Description . . . . .	643
8.11.2	Macro Definition Documentation . . . . .	643
8.11.2.1	VTSS_FDMA_CH_CNT . . . . .	644
8.11.2.2	VTSS_PHYS_PORT_CNT . . . . .	644
8.11.2.3	VTSS_FDMA_DCB_SIZE_BYTES . . . . .	644

8.11.2.4	VTSS_FDMA_HDR_SIZE_BYTES . . . . .	644
8.11.2.5	VTSS_FDMA_MAX_DATA_PER_DCB_BYTES . . . . .	644
8.11.2.6	VTSS_FDMA_MIN_FRAME_SIZE_BYTES . . . . .	645
8.11.2.7	VTSS_FDMA_MIN_DATA_PER_INJ_SOF_DCB_BYTES . . . . .	645
8.11.2.8	VTSS_FDMA_MIN_DATA_PER_XTR_SOF_DCB_BYTES . . . . .	645
8.11.2.9	VTSS_FDMA_MIN_DATA_PER_NON_SOF_DCB_BYTES . . . . .	645
8.11.2.10	VTSS_FDMA_MAX_FRAME_SIZE_BYTES . . . . .	645
8.11.2.11	VTSS_OS_DCACHE_LINE_SIZE_BYTES . . . . .	646
8.11.2.12	VTSS_OS_COMPILER_ATTRIBUTE_ALIGNED . . . . .	646
8.11.2.13	VTSS_AFI_FPS_MAX . . . . .	646
8.11.2.14	VTSS_FDMA_CCM_QUOTIENT_MAX . . . . .	646
8.11.2.15	VTSS_FDMA_CCM_FREQ_LIST_LEN . . . . .	646
8.11.2.16	VTSS_FDMA_CCM_FPS_MAX . . . . .	647
8.11.3	Typedef Documentation . . . . .	647
8.11.3.1	vtss_fdma_ch_t . . . . .	647
8.11.3.2	vtss_fdma_list_t . . . . .	649
8.11.4	Enumeration Type Documentation . . . . .	649
8.11.4.1	vtss_fdma_afi_type_t . . . . .	649
8.11.4.2	vtss_fdma_ch_usage_t . . . . .	650
8.11.4.3	vtss_fdma_dcb_type_t . . . . .	650
8.11.5	Function Documentation . . . . .	650
8.11.5.1	vtss_fdma_uninit() . . . . .	651
8.11.5.2	vtss_fdma_cfg() . . . . .	651
8.11.5.3	vtss_fdma_dcb_release() . . . . .	652
8.11.5.4	vtss_fdma_tx() . . . . .	652
8.11.5.5	vtss_fdma_tx_info_init() . . . . .	653
8.11.5.6	vtss_fdma_afi_cancel() . . . . .	654
8.11.5.7	vtss_fdma_afi_frm_cnt() . . . . .	654
8.11.5.8	vtss_fdma_dcb_get() . . . . .	655
8.11.5.9	vtss_fdma_throttle_cfg_get() . . . . .	656

8.11.5.10 <code>vtss_fdma_throttle_cfg_set()</code>	656
8.11.5.11 <code>vtss_fdma_throttle_tick()</code>	657
8.11.5.12 <code>vtss_fdma_stats_clr()</code>	657
8.11.5.13 <code>vtss_fdma_irq_handler()</code>	658
8.12 <code>vtss_api/include/vtss_gfp_api.h</code> File Reference	659
8.12.1 Detailed Description	659
8.13 <code>vtss_api/include/vtss_hqos_api.h</code> File Reference	659
8.13.1 Detailed Description	659
8.14 <code>vtss_api/include/vtss_i2c_api.h</code> File Reference	659
8.14.1 Detailed Description	660
8.15 <code>vtss_api/include/vtss_init_api.h</code> File Reference	660
8.15.1 Detailed Description	662
8.15.2 Macro Definition Documentation	662
8.15.2.1 <code>VTSS_I2C_NO_MULTIPLEXER</code>	662
8.15.2.2 <code>VTSS_QS_CONF_MAX</code>	663
8.15.2.3 <code>VTSS_QS_CONF_MIN</code>	663
8.15.3 Typedef Documentation	663
8.15.3.1 <code>vtss_reg_read_t</code>	663
8.15.3.2 <code>vtss_reg_write_t</code>	663
8.15.3.3 <code>vtss_i2c_read_t</code>	664
8.15.3.4 <code>vtss_i2c_write_t</code>	664
8.15.3.5 <code>vtss_spi_read_write_t</code>	665
8.15.3.6 <code>vtss_spi_32bit_read_write_t</code>	665
8.15.3.7 <code>vtss_spi_64bit_read_write_t</code>	666
8.15.3.8 <code>vtss_miim_read_t</code>	666
8.15.3.9 <code>vtss_miim_write_t</code>	667
8.15.3.10 <code>vtss_mmd_read_t</code>	667
8.15.3.11 <code>vtss_mmd_read_inc_t</code>	668
8.15.3.12 <code>vtss_mmd_write_t</code>	668
8.15.4 Enumeration Type Documentation	669

8.15.4.1 <code>vtss_target_type_t</code>	669
8.15.4.2 <code>vtss_port_mux_mode_t</code>	670
8.15.4.3 <code>vtss_qs_mode_t</code>	670
8.15.4.4 <code>vtss_restart_t</code>	670
8.15.5 Function Documentation	671
8.15.5.1 <code>vtss_inst_get()</code>	671
8.15.5.2 <code>vtss_inst_create()</code>	671
8.15.5.3 <code>vtss_inst_destroy()</code>	671
8.15.5.4 <code>vtss_init_conf_get()</code>	673
8.15.5.5 <code>vtss_init_conf_set()</code>	673
8.15.5.6 <code>vtss_restart_conf_end()</code>	674
8.15.5.7 <code>vtss_restart_status_get()</code>	674
8.15.5.8 <code>vtss_restart_conf_get()</code>	674
8.15.5.9 <code>vtss_restart_conf_set()</code>	675
8.15.5.10 <code>vtss_qs_conf_set()</code>	675
8.15.5.11 <code>vtss_qs_conf_get()</code>	675
8.16 <code>vtss_api/include/vtss_l2_api.h</code> File Reference	676
8.16.1 Detailed Description	684
8.16.2 Macro Definition Documentation	684
8.16.2.1 <code>VTSS_MAC_ADDRS</code>	684
8.16.2.2 <code>VTSS_VSTAX_UPSIDS</code>	685
8.16.2.3 <code>VTSS_VSTAX_UPSID_START</code>	685
8.16.2.4 <code>VTSS_VSTAX_UPSID_MIN</code>	685
8.16.2.5 <code>VTSS_VSTAX_UPSID_MAX</code>	685
8.16.2.6 <code>VTSS_VSTAX_UPSID_LEGAL</code>	685
8.16.2.7 <code>VTSS_VSTAX_UPSID_UNDEF</code>	686
8.16.2.8 <code>VTSS_UPSPN_CPU</code>	686
8.16.2.9 <code>VTSS_UPSPN_NONE</code>	686
8.16.2.10 <code>VTSS_MSTIS</code>	686
8.16.2.11 <code>VTSS_MSTI_START</code>	686

---

8.16.2.12 VTSS_MSTI_END . . . . .	687
8.16.2.13 VTSS_MSTI_ARRAY_SIZE . . . . .	687
8.16.2.14 VTSS_VCL_IDS . . . . .	687
8.16.2.15 VTSS_VCL_ID_START . . . . .	687
8.16.2.16 VTSS_VCL_ID_END . . . . .	687
8.16.2.17 VTSS_VCL_ARRAY_SIZE . . . . .	688
8.16.2.18 VTSS_VCE_ID_LAST . . . . .	688
8.16.2.19 VTSS_VLAN_TRANS_GROUP_MAX_CNT . . . . .	688
8.16.2.20 VTSS_VLAN_TRANS_MAX_CNT . . . . .	688
8.16.2.21 VTSS_VLAN_TRANS_NULL_GROUP_ID . . . . .	688
8.16.2.22 VTSS_VLAN_TRANS_FIRST_GROUP_ID . . . . .	689
8.16.2.23 VTSS_VLAN_TRANS_VID_START . . . . .	689
8.16.2.24 VTSS_VLAN_TRANS_MAX_VLAN_ID . . . . .	689
8.16.2.25 VTSS_VLAN_TRANS_LAST_GROUP_ID . . . . .	689
8.16.2.26 VTSS_VLAN_TRANS_VALID_GROUP_CHECK . . . . .	689
8.16.2.27 VTSS_VLAN_TRANS_VALID_VLAN_CHECK . . . . .	690
8.16.2.28 VTSS_VLAN_TRANS_NULL_CHECK . . . . .	690
8.16.2.29 VTSS_VLAN_TRANS_PORT_BF_SIZE . . . . .	690
8.16.2.30 VTSS_PVLANS . . . . .	690
8.16.2.31 VTSS_PVLAN_NO_START . . . . .	691
8.16.2.32 VTSS_PVLAN_NO_END . . . . .	691
8.16.2.33 VTSS_PVLAN_ARRAY_SIZE . . . . .	691
8.16.2.34 VTSS_PVLAN_NO_DEFAULT . . . . .	691
8.16.2.35 VTSS_ERPIS . . . . .	691
8.16.2.36 VTSS_ERPI_START . . . . .	692
8.16.2.37 VTSS_ERPI_END . . . . .	692
8.16.2.38 VTSS_ERPI_ARRAY_SIZE . . . . .	692
8.16.3 Typedef Documentation . . . . .	692
8.16.3.1 vtss_vt_id_t . . . . .	692
8.16.4 Enumeration Type Documentation . . . . .	692

8.16.4.1 <code>vtss_stp_state_t</code>	692
8.16.4.2 <code>vtss_vlan_port_type_t</code>	693
8.16.4.3 <code>vtss_vlan_tx_tag_t</code>	693
8.16.4.4 <code>vtss_vce_type_t</code>	693
8.16.4.5 <code>vtss_mirror_tag_t</code>	694
8.16.4.6 <code>vtss_eps_port_type_t</code>	694
8.16.4.7 <code>vtss_eps_selector_t</code>	694
8.16.4.8 <code>vtss_erps_state_t</code>	695
8.16.4.9 <code>vtss_vstax_topology_type_t</code>	695
8.16.5 Function Documentation	695
8.16.5.1 <code>vtss_mac_table_add()</code>	695
8.16.5.2 <code>vtss_mac_table_del()</code>	697
8.16.5.3 <code>vtss_mac_table_get()</code>	697
8.16.5.4 <code>vtss_mac_table_get_next()</code>	698
8.16.5.5 <code>vtss_mac_table_age_time_get()</code>	698
8.16.5.6 <code>vtss_mac_table_age_time_set()</code>	698
8.16.5.7 <code>vtss_mac_table_age()</code>	699
8.16.5.8 <code>vtss_mac_table_vlan_age()</code>	699
8.16.5.9 <code>vtss_mac_table_flush()</code>	700
8.16.5.10 <code>vtss_mac_table_port_flush()</code>	700
8.16.5.11 <code>vtss_mac_table_vlan_flush()</code>	700
8.16.5.12 <code>vtss_mac_table_vlan_port_flush()</code>	701
8.16.5.13 <code>vtss_mac_table_upsid_flush()</code>	701
8.16.5.14 <code>vtss_mac_table_upsid_upspn_flush()</code>	701
8.16.5.15 <code>vtss_mac_table_glag_add()</code>	702
8.16.5.16 <code>vtss_mac_table_glag_flush()</code>	702
8.16.5.17 <code>vtss_mac_table_vlan_glag_flush()</code>	703
8.16.5.18 <code>vtss_mac_table_status_get()</code>	703
8.16.5.19 <code>vtss_learn_port_mode_get()</code>	703
8.16.5.20 <code>vtss_learn_port_mode_set()</code>	704

8.16.5.21 vtss_port_state_get()	704
8.16.5.22 vtss_port_state_set()	705
8.16.5.23 vtss_stp_port_state_get()	705
8.16.5.24 vtss_stp_port_state_set()	706
8.16.5.25 vtss_mstp_vlan_msti_get()	706
8.16.5.26 vtss_mstp_vlan_msti_set()	706
8.16.5.27 vtss_mstp_port_msti_state_get()	707
8.16.5.28 vtss_mstp_port_msti_state_set()	707
8.16.5.29 vtss_vlan_conf_get()	708
8.16.5.30 vtss_vlan_conf_set()	708
8.16.5.31 vtss_vlan_port_conf_get()	708
8.16.5.32 vtss_vlan_port_conf_set()	710
8.16.5.33 vtss_vlan_port_members_get()	710
8.16.5.34 vtss_vlan_port_members_set()	711
8.16.5.35 vtss_vlan_vid_conf_get()	711
8.16.5.36 vtss_vlan_vid_conf_set()	712
8.16.5.37 vtss_vlan_tx_tag_get()	712
8.16.5.38 vtss_vlan_tx_tag_set()	712
8.16.5.39 vtss_vcl_port_conf_get()	713
8.16.5.40 vtss_vcl_port_conf_set()	713
8.16.5.41 vtss_vce_init()	714
8.16.5.42 vtss_vce_add()	714
8.16.5.43 vtss_vce_del()	715
8.16.5.44 vtss_vlan_trans_group_add()	715
8.16.5.45 vtss_vlan_trans_group_del()	715
8.16.5.46 vtss_vlan_trans_group_get()	716
8.16.5.47 vtss_vlan_trans_group_to_port_set()	716
8.16.5.48 vtss_vlan_trans_group_to_port_get()	717
8.16.5.49 vtss_isolated_vlan_get()	717
8.16.5.50 vtss_isolated_vlan_set()	718

8.16.5.51 vtss_isolated_port_members_get()	718
8.16.5.52 vtss_isolated_port_members_set()	718
8.16.5.53 vtss_pvlan_port_members_get()	719
8.16.5.54 vtss_pvlan_port_members_set()	719
8.16.5.55 vtss_apvlan_port_members_get()	720
8.16.5.56 vtss_apvlan_port_members_set()	720
8.16.5.57 vtss_dgroup_port_conf_get()	721
8.16.5.58 vtss_dgroup_port_conf_set()	721
8.16.5.59 vtss_sflow_port_conf_get()	721
8.16.5.60 vtss_sflow_port_conf_set()	722
8.16.5.61 vtss_sflow_sampling_rate_convert()	722
8.16.5.62 vtss_aggr_port_members_get()	723
8.16.5.63 vtss_aggr_port_members_set()	723
8.16.5.64 vtss_aggr_mode_get()	724
8.16.5.65 vtss_aggr_mode_set()	724
8.16.5.66 vtss_aggr_glag_members_get()	724
8.16.5.67 vtss_vstax_glag_get()	726
8.16.5.68 vtss_vstax_glag_set()	726
8.16.5.69 vtss_mirror_conf_get()	727
8.16.5.70 vtss_mirror_conf_set()	727
8.16.5.71 vtss_mirror_monitor_port_get()	727
8.16.5.72 vtss_mirror_monitor_port_set()	728
8.16.5.73 vtss_mirror_ingress_ports_get()	728
8.16.5.74 vtss_mirror_ingress_ports_set()	729
8.16.5.75 vtss_mirror_egress_ports_get()	729
8.16.5.76 vtss_mirror_egress_ports_set()	729
8.16.5.77 vtss_mirror_cpu_ingress_get()	730
8.16.5.78 vtss_mirror_cpu_ingress_set()	730
8.16.5.79 vtss_mirror_cpu_egress_get()	730
8.16.5.80 vtss_mirror_cpu_egress_set()	731

8.16.5.81 vtss_uc_flood_members_get()	731
8.16.5.82 vtss_uc_flood_members_set()	732
8.16.5.83 vtss_mc_flood_members_get()	732
8.16.5.84 vtss_mc_flood_members_set()	732
8.16.5.85 vtss_ipv4_mc_flood_members_get()	733
8.16.5.86 vtss_ipv4_mc_flood_members_set()	733
8.16.5.87 vtss_ipv6_mc_flood_members_get()	734
8.16.5.88 vtss_ipv6_mc_flood_members_set()	734
8.16.5.89 vtss_ipv6_mc_ctrl_flood_get()	734
8.16.5.90 vtss_ipv6_mc_ctrl_flood_set()	735
8.16.5.91 vtss_eps_port_conf_get()	735
8.16.5.92 vtss_eps_port_conf_set()	736
8.16.5.93 vtss_eps_port_selector_get()	736
8.16.5.94 vtss_eps_port_selector_set()	736
8.16.5.95 vtss_erps_vlan_member_get()	737
8.16.5.96 vtss_erps_vlan_member_set()	737
8.16.5.97 vtss_erps_port_state_get()	738
8.16.5.98 vtss_erps_port_state_set()	738
8.16.5.99 vtss_vstax_conf_get()	739
8.16.5.100 vtss_vstax_conf_set()	739
8.16.5.101 vtss_vstax_port_conf_get()	739
8.16.5.102 vtss_vstax_port_conf_set()	740
8.16.5.103 vtss_vstax_master_upsid_get()	740
8.16.5.104 vtss_vstax_master_upsid_set()	741
8.16.5.105 vtss_vstax_topology_set()	741
8.17 vtss_api/include/vtss_I3_api.h File Reference	741
8.17.1 Detailed Description	743
8.17.2 Macro Definition Documentation	743
8.17.2.1 VTSS_JR1_LPM_CNT	744
8.17.2.2 VTSS_LPM_CNT	744

8.17.2.3 VTSS_JR1_ARP_CNT . . . . .	744
8.17.2.4 VTSS_ARP_CNT . . . . .	744
8.17.2.5 VTSS_JR1_RLEG_CNT . . . . .	744
8.17.2.6 VTSS_RLEG_CNT . . . . .	745
8.17.2.7 VTSS_ARP_IPV4_RELATIONS . . . . .	745
8.17.2.8 VTSS_ARP_IPV6_RELATIONS . . . . .	745
8.17.3 Enumeration Type Documentation . . . . .	745
8.17.3.1 vtss_l3_rleg_common_mode_t . . . . .	745
8.17.3.2 vtss_l3_neighbour_type_t . . . . .	746
8.17.4 Function Documentation . . . . .	746
8.17.4.1 vtss_l3_flush() . . . . .	746
8.17.4.2 vtss_l3_common_get() . . . . .	746
8.17.4.3 vtss_l3_common_set() . . . . .	747
8.17.4.4 vtss_l3_rleg_get() . . . . .	747
8.17.4.5 vtss_l3_rleg_get_specific() . . . . .	747
8.17.4.6 vtss_l3_rleg_add() . . . . .	748
8.17.4.7 vtss_l3_rleg_update() . . . . .	748
8.17.4.8 vtss_l3_rleg_del() . . . . .	749
8.17.4.9 vtss_l3_route_get() . . . . .	749
8.17.4.10 vtss_l3_route_add() . . . . .	749
8.17.4.11 vtss_l3_route_del() . . . . .	750
8.17.4.12 vtss_l3_neighbour_get() . . . . .	750
8.17.4.13 vtss_l3_neighbour_add() . . . . .	751
8.17.4.14 vtss_l3_neighbour_del() . . . . .	751
8.17.4.15 vtss_l3_counters_reset() . . . . .	751
8.17.4.16 vtss_l3_counters_system_get() . . . . .	752
8.17.4.17 vtss_l3_counters_rleg_get() . . . . .	752
8.17.4.18 vtss_l3_counters_rleg_clear() . . . . .	753
8.18 vtss_api/include/vtss_mac10g_api.h File Reference . . . . .	753
8.18.1 Detailed Description . . . . .	753

---

8.19 <a href="#">vtss_api/include/vtss_misc_api.h File Reference</a>	753
8.19.1 <a href="#">Detailed Description</a>	758
8.19.2 <a href="#">Macro Definition Documentation</a>	759
8.19.2.1 <a href="#">VTSS_OS_TIMESTAMP_TYPE</a>	759
8.19.2.2 <a href="#">VTSS_OS_TIMESTAMP</a>	759
8.19.3 <a href="#">Typedef Documentation</a>	759
8.19.3.1 <a href="#">tod_get_ns_cnt_cb_t</a>	759
8.19.4 <a href="#">Enumeration Type Documentation</a>	759
8.19.4.1 <a href="#">vtss_trace_layer_t</a>	759
8.19.4.2 <a href="#">vtss_trace_group_t</a>	760
8.19.4.3 <a href="#">vtss_trace_level_t</a>	760
8.19.4.4 <a href="#">vtss_debug_layer_t</a>	761
8.19.4.5 <a href="#">vtss_debug_group_t</a>	761
8.19.4.6 <a href="#">vtss_gpio_mode_t</a>	762
8.19.4.7 <a href="#">vtss_sgpio_mode_t</a>	763
8.19.4.8 <a href="#">vtss_sgpio_bmode_t</a>	763
8.19.4.9 <a href="#">vtss_irq_t</a>	763
8.19.4.10 <a href="#">vtss_eee_state_select_t</a>	764
8.19.5 <a href="#">Function Documentation</a>	764
8.19.5.1 <a href="#">vtss_trace_conf_get()</a>	764
8.19.5.2 <a href="#">vtss_trace_conf_set()</a>	765
8.19.5.3 <a href="#">vtss_callout_trace_printf()</a>	765
8.19.5.4 <a href="#">vtss_callout_trace_hex_dump()</a>	766
8.19.5.5 <a href="#">vtss_debug_info_get()</a>	766
8.19.5.6 <a href="#">vtss_debug_info_print()</a>	767
8.19.5.7 <a href="#">vtss_callout_lock()</a>	767
8.19.5.8 <a href="#">vtss_callout_unlock()</a>	767
8.19.5.9 <a href="#">vtss_debug_lock()</a>	768
8.19.5.10 <a href="#">vtss_debug_unlock()</a>	768
8.19.5.11 <a href="#">vtss_reg_read()</a>	768

8.19.5.12 vtss_reg_write()	769
8.19.5.13 vtss_reg_write_masked()	769
8.19.5.14 vtss_intr_sticky_clear()	770
8.19.5.15 vtss_chip_id_get()	770
8.19.5.16 vtss_poll_1sec()	771
8.19.5.17 vtss_ptp_event_poll()	771
8.19.5.18 vtss_ptp_event_enable()	772
8.19.5.19 vtss_dev_all_event_poll()	772
8.19.5.20 vtss_dev_all_event_enable()	773
8.19.5.21 vtss_gpio_mode_set()	773
8.19.5.22 vtss_gpio_direction_set()	773
8.19.5.23 vtss_gpio_read()	774
8.19.5.24 vtss_gpio_write()	774
8.19.5.25 vtss_gpio_event_poll()	775
8.19.5.26 vtss_gpio_event_enable()	775
8.19.5.27 vtss_sgpio_conf_get()	776
8.19.5.28 vtss_sgpio_conf_set()	776
8.19.5.29 vtss_sgpio_read()	777
8.19.5.30 vtss_sgpio_event_poll()	777
8.19.5.31 vtss_sgpio_event_enable()	778
8.19.5.32 vtss_intr_cfg()	778
8.19.5.33 vtss_irq_conf_get()	779
8.19.5.34 vtss_irq_conf_set()	779
8.19.5.35 vtss_irq_status_get_and_mask()	779
8.19.5.36 vtss_irq_enable()	780
8.19.5.37 vtss_tod_get_ns_cnt()	780
8.19.5.38 vtss_tod_set_ns_cnt_cb()	780
8.19.5.39 vtss_temp_sensor_init()	781
8.19.5.40 vtss_temp_sensor_get()	781
8.19.5.41 vtss_fan_rotation_get()	781

8.19.5.42 vtss_fan_cool_lvl_set()	782
8.19.5.43 vtss_fan_controller_init()	782
8.19.5.44 vtss_fan_cool_lvl_get()	783
8.19.5.45 vtss_eee_port_conf_set()	783
8.19.5.46 vtss_eee_port_state_set()	783
8.19.5.47 vtss_eee_port_counter_get()	784
8.19.5.48 vtss_debug_reg_check_set()	784
8.20 vtss_api/include/vtss_mpls_api.h File Reference	785
8.20.1 Detailed Description	785
8.21 vtss_api/include/vtss_oam_api.h File Reference	785
8.21.1 Detailed Description	786
8.22 vtss_api/include/vtss_oha_api.h File Reference	786
8.22.1 Detailed Description	786
8.23 vtss_api/include/vtss_os.h File Reference	786
8.23.1 Detailed Description	786
8.24 vtss_api/include/vtss_os_custom.h File Reference	786
8.24.1 Detailed Description	787
8.24.2 Macro Definition Documentation	787
8.24.2.1 uint	787
8.24.2.2 ulong	787
8.24.2.3 VTSS_MSLEEP	788
8.24.2.4 VTSS_MTIMER_START	788
8.24.2.5 VTSS_MTIMER_TIMEOUT	788
8.24.2.6 VTSS_MTIMER_CANCEL	788
8.24.2.7 VTSS_DIV64	788
8.24.2.8 VTSS_MOD64	789
8.24.2.9 VTSS_LABS	789
8.24.2.10 VTSS_LLabs	789
8.24.2.11 VTSS_OS_Ctz	789
8.24.2.12 VTSS_OS_Ctz64	790

8.24.2.13 VTSS_OS_MALLOC . . . . .	790
8.24.2.14 VTSS_OS_FREE . . . . .	790
8.24.2.15 VTSS_OS_RAND . . . . .	790
8.24.3 Typedef Documentation . . . . .	791
8.24.3.1 vtss_mtimer_t . . . . .	791
8.25 vtss_api/include/vtss_os_ecos.h File Reference . . . . .	791
8.25.1 Detailed Description . . . . .	792
8.25.2 Macro Definition Documentation . . . . .	792
8.25.2.1 VTSS_MSLEEP . . . . .	792
8.25.2.2 VTSS_NSLEEP . . . . .	793
8.25.2.3 VTSS_MTIMER_START . . . . .	793
8.25.2.4 VTSS_MTIMER_TIMEOUT . . . . .	793
8.25.2.5 VTSS_MTIMER_CANCEL . . . . .	793
8.25.2.6 VTSS_TIME_OF_DAY . . . . .	793
8.25.2.7 VTSS_DIV64 . . . . .	794
8.25.2.8 VTSS_MOD64 . . . . .	794
8.25.2.9 VTSS_LABS . . . . .	794
8.25.2.10 VTSS_LLabs . . . . .	794
8.25.2.11 VTSS_OS_CTZ . . . . .	795
8.25.2.12 VTSS_OS_CTZ64 . . . . .	795
8.25.2.13 VTSS_OS_MALLOC . . . . .	795
8.25.2.14 VTSS_OS_FREE . . . . .	796
8.25.2.15 VTSS_OS_RAND . . . . .	796
8.25.2.16 VTSS_OS_REORDER_BARRIER . . . . .	796
8.25.2.17 VTSS_OS_COMPILER_ATTRIBUTE_ALIGNED . . . . .	796
8.25.2.18 VTSS_OS_DCACHE_LINE_SIZE_BYTES . . . . .	797
8.25.2.19 VTSS_OS_DCACHE_INVALIDATE . . . . .	797
8.25.2.20 VTSS_OS_DCACHE_FLUSH . . . . .	797
8.25.2.21 VTSS_OS_VIRT_TO_PHYS . . . . .	797
8.25.2.22 VTSS_OS_BIG_ENDIAN . . . . .	798

8.25.2.23 VTSS_OS_NTOHL . . . . .	798
8.25.2.24 VTSS_OS_SCHEDULER_FLAGS . . . . .	798
8.25.2.25 VTSS_OS_SCHEDULER_LOCK . . . . .	798
8.25.2.26 VTSS_OS_SCHEDULER_UNLOCK . . . . .	799
8.25.2.27 VTSS_OS_INTERRUPT_FLAGS . . . . .	799
8.25.2.28 VTSS_OS_INTERRUPT_DISABLE . . . . .	799
8.25.2.29 VTSS_OS_INTERRUPT_RESTORE . . . . .	799
8.25.3 Typedef Documentation . . . . .	799
8.25.3.1 vtss_mtimer_t . . . . .	800
8.25.4 Function Documentation . . . . .	800
8.25.4.1 llabs() . . . . .	800
8.25.4.2 vtss_callout_malloc() . . . . .	800
8.25.4.3 vtss_callout_free() . . . . .	801
8.26 vtss_api/include/vtss_os_linux.h File Reference . . . . .	801
8.26.1 Detailed Description . . . . .	802
8.26.2 Macro Definition Documentation . . . . .	802
8.26.2.1 VTSS_OS_BIG_ENDIAN . . . . .	802
8.26.2.2 VTSS_OS_NTOHL . . . . .	802
8.26.2.3 VTSS_NSLEEP . . . . .	802
8.26.2.4 VTSS_MSLEEP . . . . .	803
8.26.2.5 VTSS_MTIMER_START . . . . .	803
8.26.2.6 VTSS_MTIMER_TIMEOUT . . . . .	804
8.26.2.7 VTSS_MTIMER_CANCEL . . . . .	804
8.26.2.8 VTSS_TIME_OF_DAY . . . . .	804
8.26.2.9 VTSS_OS_SCHEDULER_FLAGS . . . . .	804
8.26.2.10 VTSS_OS_SCHEDULER_LOCK . . . . .	805
8.26.2.11 VTSS_OS_SCHEDULER_UNLOCK . . . . .	805
8.26.2.12 VTSS_DIV64 . . . . .	805
8.26.2.13 VTSS_MOD64 . . . . .	805
8.26.2.14 VTSS_LABS . . . . .	806

8.26.2.15 VTSS_LLABS . . . . .	806
8.26.2.16 VTSS_OS_CTZ . . . . .	806
8.26.2.17 VTSS_OS_CTZ64 . . . . .	807
8.26.2.18 VTSS_OS_MALLOC . . . . .	807
8.26.2.19 VTSS_OS_FREE . . . . .	807
8.26.2.20 VTSS_OS_RAND . . . . .	808
8.27 vtss_api/include/vtss_otn_api.h File Reference . . . . .	808
8.27.1 Detailed Description . . . . .	808
8.28 vtss_api/include/vtss_packet_api.h File Reference . . . . .	808
8.28.1 Detailed Description . . . . .	812
8.28.2 Macro Definition Documentation . . . . .	812
8.28.2.1 VTSS_PRIO_SUPER . . . . .	812
8.28.2.2 VTSS_VSTAX_TTL_PORT . . . . .	812
8.28.2.3 VTSS_VSTAX_HDR_SIZE . . . . .	812
8.28.2.4 VTSS_JR1_PACKET_HDR_SIZE_BYTES . . . . .	812
8.28.2.5 VTSS_JR2_PACKET_HDR_SIZE_BYTES . . . . .	813
8.28.2.6 VTSS_SVL_PACKET_HDR_SIZE_BYTES . . . . .	813
8.28.2.7 VTSS_L26_PACKET_HDR_SIZE_BYTES . . . . .	813
8.28.2.8 VTSS_PACKET_HDR_SIZE_BYTES . . . . .	813
8.28.2.9 VTSS_SVL_RX_IFH_SIZE . . . . .	813
8.28.2.10 VTSS_JR1_RX_IFH_SIZE . . . . .	814
8.28.2.11 VTSS_L26_RX_IFH_SIZE . . . . .	814
8.28.2.12 VTSS_JR2_RX_IFH_SIZE . . . . .	814
8.28.2.13 VTSS_PACKET_TX_IFH_MAX . . . . .	814
8.28.3 Enumeration Type Documentation . . . . .	814
8.28.3.1 vtss_packet_filter_t . . . . .	814
8.28.3.2 vtss_vstax_fwd_mode_t . . . . .	815
8.28.3.3 vtss_packet_oam_type_t . . . . .	815
8.28.3.4 vtss_packet_pip_action_t . . . . .	816
8.28.3.5 vtss_tag_type_t . . . . .	816

8.28.3.6 vtss_packet_rx_hints_t . . . . .	816
8.28.3.7 vtss_packet_tx_vstax_t . . . . .	817
8.28.4 Function Documentation . . . . .	818
8.28.4.1 vtss_npi_conf_get() . . . . .	818
8.28.4.2 vtss_npi_conf_set() . . . . .	818
8.28.4.3 vtss_packet_rx_conf_get() . . . . .	819
8.28.4.4 vtss_packet_rx_conf_set() . . . . .	819
8.28.4.5 vtss_packet_rx_port_conf_get() . . . . .	819
8.28.4.6 vtss_packet_rx_port_conf_set() . . . . .	821
8.28.4.7 vtss_packet_rx_frame_get() . . . . .	821
8.28.4.8 vtss_packet_rx_frame_get_raw() . . . . .	822
8.28.4.9 vtss_packet_rx_frame_discard() . . . . .	822
8.28.4.10 vtss_packet_tx_frame_port() . . . . .	823
8.28.4.11 vtss_packet_tx_frame_port_vlan() . . . . .	823
8.28.4.12 vtss_packet_tx_frame_vlan() . . . . .	824
8.28.4.13 vtss_packet_frame_info_init() . . . . .	824
8.28.4.14 vtss_packet_frame_filter() . . . . .	824
8.28.4.15 vtss_packet_port_info_init() . . . . .	825
8.28.4.16 vtss_packet_port_filter_get() . . . . .	825
8.28.4.17 vtss_packet_tx_frame_vstax() . . . . .	826
8.28.4.18 vtss_packet_vstax_header2frame() . . . . .	826
8.28.4.19 vtss_packet_vstax_frame2header() . . . . .	827
8.28.4.20 vtss_packet_rx_hdr_decode() . . . . .	827
8.28.4.21 vtss_packet_tx_hdr_encode() . . . . .	828
8.28.4.22 vtss_packet_tx_hdr_compile() . . . . .	828
8.28.4.23 vtss_packet_tx_frame() . . . . .	829
8.28.4.24 vtss_packet_tx_info_init() . . . . .	829
8.28.4.25 vtss_packet_dma_conf_get() . . . . .	830
8.28.4.26 vtss_packet_dma_conf_set() . . . . .	830
8.28.4.27 vtss_packet_dma_offset() . . . . .	831

8.29 vtss_api/include/vtss_pcs_10gbase_r_api.h File Reference . . . . .	831
8.29.1 Detailed Description . . . . .	831
8.30 vtss_api/include/vtss_phy_10g_api.h File Reference . . . . .	831
8.30.1 Detailed Description . . . . .	846
8.30.2 Macro Definition Documentation . . . . .	846
8.30.2.1 VTSS_SLEWRATE_25PS . . . . .	846
8.30.2.2 VTSS_SLEWRATE_35PS . . . . .	846
8.30.2.3 VTSS_SLEWRATE_55PS . . . . .	847
8.30.2.4 VTSS_SLEWRATE_70PS . . . . .	847
8.30.2.5 VTSS_SLEWRATE_120PS . . . . .	847
8.30.2.6 VTSS_SLEWRATE_INVALID . . . . .	847
8.30.2.7 BOOLEAN_STORAGE_COUNT . . . . .	847
8.30.2.8 UNSIGNED_STORAGE_COUNT . . . . .	848
8.30.2.9 PHASE_POINTS . . . . .	848
8.30.2.10 AMPLITUDE_POINTS . . . . .	848
8.30.2.11 VTSS_PHY_10G_ONE_LINE_ACTIVE . . . . .	848
8.30.2.12 VTSS_PHY_10G_MACSEC_DISABLED . . . . .	848
8.30.2.13 VTSS_PHY_10G_TIMESTAMP_DISABLED . . . . .	849
8.30.2.14 VTSS_PHY_10G_MACSEC_KEY_128 . . . . .	849
8.30.2.15 VTSS_10G_PHY_GPIO_MAX . . . . .	849
8.30.2.16 VTSS_10G_PHY_GPIO_MAL_MAX . . . . .	849
8.30.2.17 VTSS_PHY_10G_LINK_LOS_EV . . . . .	849
8.30.2.18 VTSS_PHY_10G_RX_LOL_EV [1/2] . . . . .	850
8.30.2.19 VTSS_PHY_10G_TX_LOL_EV [1/2] . . . . .	850
8.30.2.20 VTSS_PHY_10G_LOPC_EV . . . . .	850
8.30.2.21 VTSS_PHY_10G_HIGH_BER_EV . . . . .	850
8.30.2.22 VTSS_PHY_10G_MODULE_STAT_EV . . . . .	850
8.30.2.23 VTSS_PHY_10G_PCS_RECEIVE_FAULT_EV . . . . .	851
8.30.2.24 VTSS_PHY_EWIS_SEF_EV . . . . .	851
8.30.2.25 VTSS_PHY_EWIS_FPLM_EV . . . . .	851

CONTENTS	cxv
8.30.2.26 VTSS_PHY_EWIS_FAIS_EV . . . . .	851
8.30.2.27 VTSS_PHY_EWIS_LOF_EV . . . . .	851
8.30.2.28 VTSS_PHY_EWIS_RDIL_EV . . . . .	852
8.30.2.29 VTSS_PHY_EWIS_AISL_EV . . . . .	852
8.30.2.30 VTSS_PHY_EWIS_LCDP_EV . . . . .	852
8.30.2.31 VTSS_PHY_EWIS_PLMP_EV . . . . .	852
8.30.2.32 VTSS_PHY_EWIS_AISP_EV . . . . .	852
8.30.2.33 VTSS_PHY_EWIS_LOPP_EV . . . . .	853
8.30.2.34 VTSS_PHY_EWIS_UNEQP_EV . . . . .	853
8.30.2.35 VTSS_PHY_EWIS_FEUNEQP_EV . . . . .	853
8.30.2.36 VTSS_PHY_EWIS_FERDIP_EV . . . . .	853
8.30.2.37 VTSS_PHY_EWIS_REIL_EV . . . . .	853
8.30.2.38 VTSS_PHY_EWIS_REIP_EV . . . . .	854
8.30.2.39 VTSS_PHY_EWIS_B1_NZ_EV . . . . .	854
8.30.2.40 VTSS_PHY_EWIS_B2_NZ_EV . . . . .	854
8.30.2.41 VTSS_PHY_EWIS_B3_NZ_EV . . . . .	854
8.30.2.42 VTSS_PHY_EWIS_REIL_NZ_EV . . . . .	854
8.30.2.43 VTSS_PHY_EWIS_REIP_NZ_EV . . . . .	855
8.30.2.44 VTSS_PHY_EWIS_B1_THRESH_EV . . . . .	855
8.30.2.45 VTSS_PHY_EWIS_B2_THRESH_EV . . . . .	855
8.30.2.46 VTSS_PHY_EWIS_B3_THRESH_EV . . . . .	855
8.30.2.47 VTSS_PHY_EWIS_REIL_THRESH_EV . . . . .	855
8.30.2.48 VTSS_PHY_EWIS_REIP_THRESH_EV . . . . .	856
8.30.2.49 VTSS_PHY_10G_RX_LOS_EV . . . . .	856
8.30.2.50 VTSS_PHY_10G_RX_LOL_EV [2/2] . . . . .	856
8.30.2.51 VTSS_PHY_10G_TX_LOL_EV [2/2] . . . . .	856
8.30.2.52 VTSS_PHY_10G_RX_CHAR_DEC_CNT_THRESH_EV . . . . .	856
8.30.2.53 VTSS_PHY_10G_TX_CHAR_ENC_CNT_THRESH_EV . . . . .	857
8.30.2.54 VTSS_PHY_10G_RX_BLK_DEC_CNT_THRESH_EV . . . . .	857
8.30.2.55 VTSS_PHY_10G_TX_BLK_ENC_CNT_THRESH_EV . . . . .	857

8.30.2.56 VTSS_PHY_10G_RX_SEQ_CNT_THRESH_EV . . . . .	857
8.30.2.57 VTSS_PHY_10G_TX_SEQ_CNT_THRESH_EV . . . . .	857
8.30.2.58 VTSS_PHY_10G_FEC_UNFIXED_CNT_THRESH_EV . . . . .	858
8.30.2.59 VTSS_PHY_10G_FEC_FIXED_CNT_THRESH_EV . . . . .	858
8.30.2.60 VTSS_PHY_10G_HIGHBER_EV . . . . .	858
8.30.2.61 VTSS_PHY_10G_RX_LINK_STAT_EV [1/2] . . . . .	858
8.30.2.62 VTSS_PHY_10G_RX_LINK_STAT_EV [2/2] . . . . .	858
8.30.2.63 VTSS_PHY_10G_GPIO_INT_AGG0_EV . . . . .	859
8.30.2.64 VTSS_PHY_10G_GPIO_INT_AGG1_EV . . . . .	859
8.30.2.65 VTSS_PHY_10G_GPIO_INT_AGG2_EV . . . . .	859
8.30.2.66 VTSS_PHY_10G_GPIO_INT_AGG3_EV . . . . .	859
8.30.2.67 VTSS_PHY_1G_LINE_AUTONEG_RESTART_EV . . . . .	859
8.30.2.68 VTSS_PHY_1G_HOST_AUTONEG_RESTART_EV . . . . .	860
8.30.2.69 VTSS_PHY_10G_LINE_MAC_LOCAL_FAULT_EV . . . . .	860
8.30.2.70 VTSS_PHY_10G_HOST_MAC_LOCAL_FAULT_EV . . . . .	860
8.30.2.71 VTSS_PHY_10G_LINE_MAC_REMOTE_FAULT_EV . . . . .	860
8.30.2.72 VTSS_PHY_10G_HOST_MAC_REMOTE_FAULT_EV . . . . .	860
8.30.3 Typedef Documentation . . . . .	861
8.30.3.1 vtss_gpio_10g_no_t . . . . .	861
8.30.3.2 ckout_sel_t . . . . .	861
8.30.3.3 vtss_32_cntr_t . . . . .	861
8.30.3.4 vtss_phy_10g_event_t . . . . .	861
8.30.3.5 vtss_phy_10g_extnd_event_t . . . . .	861
8.30.4 Enumeration Type Documentation . . . . .	861
8.30.4.1 oper_mode_t . . . . .	861
8.30.4.2 vtss_wrefclk_t . . . . .	862
8.30.4.3 vtss_phy_interface_mode . . . . .	863
8.30.4.4 vtss_recvrd_t . . . . .	863
8.30.4.5 vtss_recvrdclk_cdr_div_t . . . . .	863
8.30.4.6 vtss_srefclk_div_t . . . . .	864

8.30.4.7 vtss_wref_clk_div_t . . . . .	864
8.30.4.8 apc_ib_regulator_t . . . . .	864
8.30.4.9 ddr_mode_t . . . . .	865
8.30.4.10 clk_mstr_t . . . . .	865
8.30.4.11 vtss_rptr_rate_t . . . . .	865
8.30.4.12 vtss_phy_10g_media_t . . . . .	866
8.30.4.13 vtss_phy_6g_link_partner_distance_t . . . . .	866
8.30.4.14 vtss_phy_10g_ib_apc_op_mode_t . . . . .	867
8.30.4.15 vtss_channel_t . . . . .	867
8.30.4.16 vtss_recvrd_clkout_t . . . . .	868
8.30.4.17 vtss_phy_10g_srefclk_freq_t . . . . .	868
8.30.4.18 vtss_phy_10g_ckout_freq_t . . . . .	868
8.30.4.19 vtss_ckout_data_sel_t . . . . .	869
8.30.4.20 vtss_phy_10g_squelch_src_t . . . . .	869
8.30.4.21 vtss_phy_10g_clk_sel_t . . . . .	870
8.30.4.22 vtss_phy_10g_recvrd_clk_sel_t . . . . .	871
8.30.4.23 ckout_sel_ . . . . .	871
8.30.4.24 vtss_phy_10g_sckout_freq_t . . . . .	872
8.30.4.25 vtss_phy_10g_rx_macro_t . . . . .	872
8.30.4.26 vtss_phy_10g_tx_macro_t . . . . .	872
8.30.4.27 vtss_phy_10g_clause_37_remote_fault_t . . . . .	873
8.30.4.28 vtss_lb_type_t . . . . .	873
8.30.4.29 vtss_phy_10g_power_t . . . . .	874
8.30.4.30 vtss_phy_10g_failover_mode_t . . . . .	874
8.30.4.31 vtss_phy_10g_auto_failover_event_t . . . . .	875
8.30.4.32 vtss_phy_10g_auto_failover_filter_t . . . . .	875
8.30.4.33 vtss_phy_10g_vsphere_scan_t . . . . .	876
8.30.4.34 vtss_phy_10g_pkt_mon_RST_t . . . . .	876
8.30.4.35 vtss_10g_phy_gpio_t . . . . .	876
8.30.4.36 vtss_gpio_10g_gpio_intr_sgnl_t . . . . .	877

8.30.4.37 vtss_gpio_10g_chan_intrpt_t . . . . .	880
8.30.4.38 vtss_gpio_10g_aggr_intrpt_t . . . . .	880
8.30.4.39 vtss_gpio_10g_input_t . . . . .	881
8.30.4.40 vtss_gpio_10g_led_mode_t . . . . .	881
8.30.4.41 vtss_gpio_10g_led_blink_t . . . . .	882
8.30.5 Function Documentation . . . . .	882
8.30.5.1 vtss_phy_10g_mode_get() . . . . .	882
8.30.5.2 vtss_phy_10g_init() . . . . .	883
8.30.5.3 vtss_phy_10g_mode_set() . . . . .	883
8.30.5.4 vtss_phy_10g_ib_conf_set() . . . . .	884
8.30.5.5 vtss_phy_10g_ib_conf_get() . . . . .	884
8.30.5.6 vtss_phy_10g_ib_status_get() . . . . .	885
8.30.5.7 vtss_phy_10g_apc_conf_set() . . . . .	885
8.30.5.8 vtss_phy_10g_apc_conf_get() . . . . .	885
8.30.5.9 vtss_phy_10g_apc_status_get() . . . . .	886
8.30.5.10 vtss_phy_10g_apc_restart() . . . . .	886
8.30.5.11 vtss_phy_10g_jitter_conf_set() . . . . .	887
8.30.5.12 vtss_phy_10g_jitter_conf_get() . . . . .	887
8.30.5.13 vtss_phy_10g_jitter_status_get() . . . . .	888
8.30.5.14 vtss_phy_10g_syncce_clkout_get() . . . . .	888
8.30.5.15 vtss_phy_10g_syncce_clkout_set() . . . . .	889
8.30.5.16 vtss_phy_10g_xfp_clkout_get() . . . . .	889
8.30.5.17 vtss_phy_10g_xfp_clkout_set() . . . . .	890
8.30.5.18 vtss_phy_10g_rxckout_get() . . . . .	890
8.30.5.19 vtss_phy_10g_rxckout_set() . . . . .	891
8.30.5.20 vtss_phy_10g_txckout_get() . . . . .	891
8.30.5.21 vtss_phy_10g_txckout_set() . . . . .	891
8.30.5.22 vtss_phy_10g_srefclk_conf_get() . . . . .	892
8.30.5.23 vtss_phy_10g_srefclk_conf_set() . . . . .	892
8.30.5.24 vtss_phy_10g_sckout_conf_set() . . . . .	893

8.30.5.25 vtss_phy_10g_ckout_conf_set()	893
8.30.5.26 vtss_phy_10g_line_clk_conf_set()	894
8.30.5.27 vtss_phy_10g_host_clk_conf_set()	894
8.30.5.28 vtss_phy_10g_line_recvrd_clk_conf_set()	895
8.30.5.29 vtss_phy_10g_host_recvrd_clk_conf_set()	895
8.30.5.30 vtss_phy_10g_lane_sync_set()	896
8.30.5.31 vtss_phy_10g_debug_register_dump()	896
8.30.5.32 vtss_phy_10g_base_kr_train_aneg_get()	897
8.30.5.33 vtss_phy_10g_base_kr_train_aneg_set()	897
8.30.5.34 vtss_phy_10g_base_host_kr_train_aneg_set()	897
8.30.5.35 vtss_phy_10g_base_kr_conf_get()	898
8.30.5.36 vtss_phy_10g_base_kr_conf_set()	898
8.30.5.37 vtss_phy_10g_base_kr_host_conf_get()	899
8.30.5.38 vtss_phy_10g_base_kr_host_conf_set()	899
8.30.5.39 vtss_phy_10g_kr_status_get()	900
8.30.5.40 vtss_phy_10g_ob_status_get()	900
8.30.5.41 vtss_phy_10g_status_get()	901
8.30.5.42 vtss_phy_10g_serdes_status_get()	901
8.30.5.43 vtss_phy_10g_reset()	902
8.30.5.44 vtss_phy_10g_clause_37_status_get()	902
8.30.5.45 vtss_phy_10g_clause_37_control_get()	902
8.30.5.46 vtss_phy_10g_clause_37_control_set()	903
8.30.5.47 vtss_phy_10g_loopback_set()	903
8.30.5.48 vtss_phy_10g_loopback_get()	904
8.30.5.49 vtss_phy_10g_cnt_get()	904
8.30.5.50 vtss_phy_10g_power_get()	905
8.30.5.51 vtss_phy_10g_power_set()	905
8.30.5.52 vtss_phy_10G_is_valid()	906
8.30.5.53 vtss_phy_10g_failover_set()	906
8.30.5.54 vtss_phy_10g_failover_get()	906

8.30.5.55 vtss_phy_10g_auto_failover_set()	907
8.30.5.56 vtss_phy_10g_auto_failover_get()	907
8.30.5.57 vtss_phy_10g_vscope_conf_set()	908
8.30.5.58 vtss_phy_10g_vscope_conf_get()	908
8.30.5.59 vtss_phy_10g_vscope_scan_status_get()	909
8.30.5.60 vtss_phy_10g_pcs_prbs_gen_conf_set()	909
8.30.5.61 vtss_phy_10g_pcs_prbs_gen_conf_get()	910
8.30.5.62 vtss_phy_10g_pcs_prbs_mon_conf_set()	910
8.30.5.63 vtss_phy_10g_pcs_prbs_mon_conf_get()	911
8.30.5.64 vtss_phy_10g_pcs_prbs_mon_status_get()	911
8.30.5.65 vtss_phy_10g_prbs_gen_conf()	912
8.30.5.66 vtss_phy_10g_prbs_gen_conf_get()	912
8.30.5.67 vtss_phy_10g_prbs_mon_conf()	912
8.30.5.68 vtss_phy_10g_prbs_mon_conf_get()	913
8.30.5.69 vtss_phy_10g_prbs_mon_status_get()	913
8.30.5.70 vtss_phy_10g_pkt_gen_conf()	914
8.30.5.71 vtss_phy_10g_pkt_mon_conf()	914
8.30.5.72 vtss_phy_10g_pkt_mon_counters_get()	915
8.30.5.73 vtss_phy_10g_id_get()	915
8.30.5.74 vtss_phy_10g_gpio_mode_set()	916
8.30.5.75 vtss_phy_10g_gpio_mode_get()	916
8.30.5.76 vtss_phy_10g_gpio_read()	917
8.30.5.77 vtss_phy_10g_gpio_write()	917
8.30.5.78 vtss_phy_10g_event_enable_set()	918
8.30.5.79 vtss_phy_10g_event_enable_get()	918
8.30.5.80 vtss_phy_10g_extended_event_enable_get()	918
8.30.5.81 vtss_phy_10g_event_poll()	919
8.30.5.82 vtss_phy_10g_pcs_status_get()	919
8.30.5.83 vtss_phy_10g_extended_event_poll()	920
8.30.5.84 vtss_phy_10g_extended_event_enable_set()	920

8.30.5.85 vtss_phy_10g_poll_1sec()	921
8.30.5.86 vtss_phy_10g_edc_fw_status_get()	921
8.30.5.87 vtss_phy_10g_fc_buffer_reset()	921
8.30.5.88 vtss_phy_10g_csr_read()	922
8.30.5.89 vtss_phy_10g_csr_write()	922
8.30.5.90 vtss_phy_warm_start_10g_failed_get()	923
8.30.5.91 vtss_phy_10g_sgmii_mode_set()	923
8.30.5.92 vtss_phy_10g_i2c_reset()	924
8.30.5.93 vtss_phy_10g_i2c_read()	924
8.30.5.94 vtss_phy_10g_i2c_write()	924
8.30.5.95 vtss_phy_10g_i2c_slave_conf_set()	925
8.30.5.96 vtss_phy_10g_i2c_slave_conf_get()	925
8.30.5.97 vtss_phy_10g_get_user_data()	926
8.31 vtss_api/include/vtss_phy_api.h File Reference	926
8.31.1 Detailed Description	936
8.31.2 Macro Definition Documentation	936
8.31.2.1 MAX_CFG_BUF_SIZE	936
8.31.2.2 MAX_STAT_BUF_SIZE	936
8.31.2.3 VTSS_PHY_POWER_ACTIPHYS_BIT	937
8.31.2.4 VTSS_PHY_POWER_DYNAMIC_BIT	937
8.31.2.5 VTSS_PHY_ACTIPHYS_PWR	937
8.31.2.6 VTSS_PHY_LINK_DOWN_PWR	937
8.31.2.7 VTSS_PHY_LINK_UP_FULL_PWR	937
8.31.2.8 VTSS_PHY_RECov_CLK1	938
8.31.2.9 VTSS_PHY_RECov_CLK2	938
8.31.2.10 VTSS_PHY_RECov_CLK_NUM	938
8.31.2.11 VTSS_PHY_PAGE_STANDARD	938
8.31.2.12 VTSS_PHY_PAGE_EXTENDED	938
8.31.2.13 VTSS_PHY_PAGE_EXTENDED_2	939
8.31.2.14 VTSS_PHY_PAGE_EXTENDED_3	939

8.31.2.15 VTSS_PHY_PAGE_EXTENDED_4 . . . . .	939
8.31.2.16 VTSS_PHY_PAGE_GPIO . . . . .	939
8.31.2.17 VTSS_PHY_PAGE_1588 . . . . .	939
8.31.2.18 VTSS_PHY_PAGE_MACSEC . . . . .	940
8.31.2.19 VTSS_PHY_PAGE_TEST . . . . .	940
8.31.2.20 VTSS_PHY_PAGE_TR . . . . .	940
8.31.2.21 VTSS_PHY_PAGE_0x2DAF . . . . .	940
8.31.2.22 VTSS_PHY_REG_STANDARD . . . . .	940
8.31.2.23 VTSS_PHY_REG_EXTENDED . . . . .	941
8.31.2.24 VTSS_PHY_REG_GPIO . . . . .	941
8.31.2.25 VTSS_PHY_REG_TEST . . . . .	941
8.31.2.26 VTSS_PHY_REG_TR . . . . .	941
8.31.2.27 VTSS_PHY_LINK_LOS_EV . . . . .	941
8.31.2.28 VTSS_PHY_LINK_FFAIL_EV . . . . .	942
8.31.2.29 VTSS_PHY_LINK_AMS_EV . . . . .	942
8.31.2.30 VTSS_PHY_LINK_SPEED_STATE_CHANGE_EV . . . . .	942
8.31.2.31 VTSS_PHY_LINK_FDX_STATE_CHANGE_EV . . . . .	942
8.31.2.32 VTSS_PHY_LINK_AUTO_NEG_ERROR_EV . . . . .	942
8.31.2.33 VTSS_PHY_LINK_AUTO_NEG_COMPLETE_EV . . . . .	943
8.31.2.34 VTSS_PHY_LINK_INLINE_POW_DEV_DETECT_EV . . . . .	943
8.31.2.35 VTSS_PHY_LINK_SYMBOL_ERR_INT_EV . . . . .	943
8.31.2.36 VTSS_PHY_LINK_TX_FIFO_OVERFLOW_INT_EV . . . . .	943
8.31.2.37 VTSS_PHY_LINK_RX_FIFO_OVERFLOW_INT_EV . . . . .	943
8.31.2.38 VTSS_PHY_LINK_FALSE_CARRIER_INT_EV . . . . .	944
8.31.2.39 VTSS_PHY_LINK_LINK_SPEED_DS_DETECT_EV . . . . .	944
8.31.2.40 VTSS_PHY_LINK_MASTER_SLAVE_RES_ERR_EV . . . . .	944
8.31.2.41 VTSS_PHY_LINK_RX_ER_INT_EV . . . . .	944
8.31.2.42 VTSS_PHY_LINK_EXTENDED_REG_INT_EV . . . . .	944
8.31.2.43 VTSS_PHY_LINK_WAKE_ON_LAN_INT_EV . . . . .	945
8.31.2.44 VTSS_PHY_LINK_EXT_EEE_WAKE_ERR_EV . . . . .	945

---

8.31.2.45 VTSS_PHY_LINK_EXT_EEE_WAIT_TS_EV . . . . .	945
8.31.2.46 VTSS_PHY_LINK_EXT_EEE_WAIT_RX_TQ_EV . . . . .	945
8.31.2.47 VTSS_PHY_LINK_EXT_EEE_LINKFAIL_EV . . . . .	945
8.31.2.48 VTSS_PHY_LINK_EXT_RR_SW_COMPL_EV . . . . .	946
8.31.2.49 VTSS_PHY_LINK_EXT_MACSEC_HOST_MAC_EV . . . . .	946
8.31.2.50 VTSS_PHY_LINK_EXT_MACSEC_LINE_MAC_EV . . . . .	946
8.31.2.51 VTSS_PHY_LINK_EXT_MACSEC_FC_BUFF_EV . . . . .	946
8.31.2.52 VTSS_PHY_LINK_EXT_MACSEC_INGRESS_EV . . . . .	946
8.31.2.53 VTSS_PHY_LINK_EXT_MACSEC_EGRESS_EV . . . . .	947
8.31.2.54 VTSS_PHY_LINK_EXT_MEM_INT_RING_EV . . . . .	947
8.31.2.55 MAX_WOL_MAC_ADDR_SIZE . . . . .	947
8.31.2.56 MAX_WOL_PASSWD_SIZE . . . . .	947
8.31.2.57 MIN_WOL_PASSWD_SIZE . . . . .	947
8.31.3 Typedef Documentation . . . . .	948
8.31.3.1 vtss_phy_recov_clk_t . . . . .	948
8.31.3.2 vtss_phy_led_intensity . . . . .	948
8.31.4 Enumeration Type Documentation . . . . .	948
8.31.4.1 vtss_phy_led_mode_t . . . . .	948
8.31.4.2 vtss_phy_media_interface_t . . . . .	949
8.31.4.3 vtss_phy_mdi_t . . . . .	950
8.31.4.4 rgmii_skew_delay_psec_t . . . . .	950
8.31.4.5 vtss_phy_forced_reset_t . . . . .	951
8.31.4.6 vtss_phy_pkt_mode_t . . . . .	951
8.31.4.7 vtss_phy_mode_t . . . . .	951
8.31.4.8 vtss_phy_fast_link_fail_t . . . . .	952
8.31.4.9 vtss_phy_sigdet_polarity_t . . . . .	952
8.31.4.10 vtss_phy_unidirectional_t . . . . .	952
8.31.4.11 vtss_phy_mac_serd_pcs_sgmii_pre . . . . .	952
8.31.4.12 vtss_phy_media_rem_fault_t . . . . .	954
8.31.4.13 vtss_phy_media_force_ams_sel_t . . . . .	954

8.31.4.14 <code>vtss_phy_clk_source_t</code>	955
8.31.4.15 <code>vtss_phy_freq_t</code>	955
8.31.4.16 <code>vtss_phy_clk_squelch</code>	955
8.31.4.17 <code>vtss_phy_gpio_mode_t</code>	956
8.31.4.18 <code>lb_type</code>	956
8.31.4.19 <code>vtss_wol_passwd_len_type_t</code>	957
8.31.4.20 <code>vtss_fefi_mode_t</code>	957
8.31.5 Function Documentation	957
8.31.5.1 <code>vtss_phy_pre_reset()</code>	957
8.31.5.2 <code>vtss_phy_post_reset()</code>	958
8.31.5.3 <code>vtss_phy_pre_system_reset()</code>	958
8.31.5.4 <code>vtss_phy_reset()</code>	959
8.31.5.5 <code>vtss_phy_reset_get()</code>	959
8.31.5.6 <code>vtss_phy_chip_temp_get()</code>	959
8.31.5.7 <code>vtss_phy_chip_temp_init()</code>	960
8.31.5.8 <code>vtss_phy_conf_get()</code>	960
8.31.5.9 <code>vtss_phy_conf_set()</code>	961
8.31.5.10 <code>vtss_phy_status_get()</code>	961
8.31.5.11 <code>vtss_phy_cl37_lp_abil_get()</code>	962
8.31.5.12 <code>vtss_phy_id_get()</code>	962
8.31.5.13 <code>vtss_phy_conf_1g_get()</code>	962
8.31.5.14 <code>vtss_phy_conf_1g_set()</code>	963
8.31.5.15 <code>vtss_phy_status_1g_get()</code>	963
8.31.5.16 <code>vtss_phy_power_conf_get()</code>	964
8.31.5.17 <code>vtss_phy_power_conf_set()</code>	964
8.31.5.18 <code>vtss_phy_power_status_get()</code>	965
8.31.5.19 <code>vtss_phy_clock_conf_set()</code>	965
8.31.5.20 <code>vtss_phy_clock_conf_get()</code>	965
8.31.5.21 <code>vtss_phy_i2c_read()</code>	966
8.31.5.22 <code>vtss_phy_i2c_write()</code>	967

8.31.5.23 vtss_phy_read()	967
8.31.5.24 vtss_phy_read_page()	968
8.31.5.25 vtss_phy_mmd_read()	968
8.31.5.26 vtss_phy_mmd_write()	969
8.31.5.27 vtss_phy_write()	969
8.31.5.28 vtss_phy_write_masked()	970
8.31.5.29 vtss_phy_write_masked_page()	970
8.31.5.30 vtss_phy_gpio_mode()	971
8.31.5.31 vtss_phy_gpio_get()	971
8.31.5.32 vtss_phy_gpio_set()	972
8.31.5.33 vtss_phy_veriphy_start()	972
8.31.5.34 vtss_phy_veriphy_get()	973
8.31.5.35 vtss_phy_led_mode_set()	973
8.31.5.36 vtss_phy_led_intensity_set()	974
8.31.5.37 vtss_phy_led_intensity_get()	974
8.31.5.38 vtss_phy_enhanced_led_control_init()	974
8.31.5.39 vtss_phy_enhanced_led_control_init_get()	975
8.31.5.40 vtss_phy_coma_mode_disable()	975
8.31.5.41 vtss_phy_coma_mode_enable()	976
8.31.5.42 vga_adc_debug()	976
8.31.5.43 vtss_phy_port_eee_capable()	976
8.31.5.44 vtss_phy_eee_ena()	977
8.31.5.45 vtss_phy_eee_conf_get()	977
8.31.5.46 vtss_phy_eee_conf_set()	978
8.31.5.47 vtss_phy_eee_power_save_state_get()	978
8.31.5.48 vtss_phy_eee_link_partner_advertisements_get()	979
8.31.5.49 vtss_phy_event_enable_set()	979
8.31.5.50 vtss_phy_event_enable_get()	980
8.31.5.51 vtss_phy_event_poll()	980
8.31.5.52 vtss_squelch_workaround()	980

8.31.5.53 vtss_phy_csr_wr()	981
8.31.5.54 vtss_phy_csr_rd()	981
8.31.5.55 vtss_phy_statistic_get()	983
8.31.5.56 vtss_phy_do_page_chk_set()	983
8.31.5.57 vtss_phy_do_page_chk_get()	984
8.31.5.58 vtss_phy_loopback_set()	984
8.31.5.59 vtss_phy_loopback_get()	984
8.31.5.60 vtss_phy_is_8051_crc_ok()	985
8.31.5.61 vtss_phy_cfg_ob_post0()	985
8.31.5.62 vtss_phy_cfg_ib_cterm()	986
8.31.5.63 vtss_phy_atom12_patch_settings_get()	986
8.31.5.64 vtss_phy_reg_decode_status()	987
8.31.5.65 vtss_phy_flowcontrol_decode_status()	987
8.31.5.66 vtss_phy_debug_stat_print()	988
8.31.5.67 vtss_phy_warm_start_failed_get()	988
8.31.5.68 vtss_phy_debug_phyinfo_print()	989
8.31.5.69 vtss_phy_debug_register_dump()	989
8.31.5.70 vtss_phy_detect_base_ports()	990
8.31.5.71 vtss_phy_ext_connector_loopback()	990
8.31.5.72 vtss_phy_serdes_sgmii_loopback()	990
8.31.5.73 vtss_phy_serdes_fmedia_loopback()	991
8.31.5.74 vtss_phy_debug_regdump_print()	991
8.31.5.75 vtss_phy_wol_enable()	992
8.31.5.76 vtss_phy_wol_conf_get()	992
8.31.5.77 vtss_phy_wol_conf_set()	993
8.31.5.78 vtss_phy_patch_settings_get()	993
8.31.5.79 vtss_phy_serdes6g_rcpll_status_get()	994
8.31.5.80 vtss_phy_serdes1g_rcpll_status_get()	994
8.31.5.81 vtss_phy_lcpll_status_get()	995
8.31.5.82 vtss_phy_reset_lcpll()	995

8.31.5.83 <code>vtss_phy_sd6g_ob_post_rd()</code>	996
8.31.5.84 <code>vtss_phy_sd6g_ob_post_wr()</code>	996
8.31.5.85 <code>vtss_phy_sd6g_ob_lev_rd()</code>	997
8.31.5.86 <code>vtss_phy_sd6g_ob_lev_wr()</code>	997
8.31.5.87 <code>vtss_phy_mac_media_inhibit_odd_start()</code>	998
8.31.5.88 <code>vtss_phy_fefi_get()</code>	998
8.31.5.89 <code>vtss_phy_fefi_set()</code>	998
8.31.5.90 <code>vtss_phy_fefi_detect()</code>	999
8.31.5.91 <code>vtss_phy_mse_100m_get()</code>	999
8.31.5.92 <code>vtss_phy_mse_1000m_get()</code>	1000
8.31.5.93 <code>vtss_phy_read_tr_addr()</code>	1000
8.31.5.94 <code>vtss_phy_is_viper_revB()</code>	1001
8.31.5.95 <code>vtss_phy_ext_event_poll()</code>	1001
8.31.5.96 <code>vtss_phy_status_inst_poll()</code>	1002
8.31.5.97 <code>vtss_phy_macsec_csr_sd6g_rd()</code>	1002
8.31.5.98 <code>vtss_phy_macsec_csr_sd6g_wr()</code>	1003
8.31.5.99 <code>vtss_phy_sd6g_mac_serdes_conf()</code>	1003
8.32 <code>vtss_api/include/vtss_phy_ts_api.h</code> File Reference	1003
8.32.1 Detailed Description	1012
8.32.2 Macro Definition Documentation	1013
8.32.2.1 <code>VTSS_PHY_TS_FIFO_SIG_SRC_IP</code>	1013
8.32.2.2 <code>VTSS_PHY_TS_FIFO_SIG_DEST_IP</code>	1013
8.32.2.3 <code>VTSS_PHY_TS_FIFO_SIG_MSG_TYPE</code>	1013
8.32.2.4 <code>VTSS_PHY_TS_FIFO_SIG_DOMAIN_NUM</code>	1013
8.32.2.5 <code>VTSS_PHY_TS_FIFO_SIG_SOURCE_PORT_ID</code>	1014
8.32.2.6 <code>VTSS_PHY_TS_FIFO_SIG_SEQ_ID</code>	1014
8.32.2.7 <code>VTSS_PHY_TS_FIFO_SIG_DEST_MAC</code>	1014
8.32.2.8 <code>VTSS_PHY_TS_SIG_LEN</code>	1014
8.32.2.9 <code>VTSS_PHY_TS_SIG_TIME_STAMP_LEN</code>	1014
8.32.2.10 <code>VTSS_PHY_TS_SIG_DOMAIN_NUM_LEN</code>	1015

8.32.2.11 VTSS_PHY_TS_SIG_SEQ_ID_LEN . . . . .	1015
8.32.2.12 VTSS_PHY_TS_SIG_MSG_TYPE_LEN . . . . .	1015
8.32.2.13 VTSS_PHY_TS_SIG_SOURCE_PORT_ID_LEN . . . . .	1015
8.32.2.14 VTSS_PHY_TS_SIG_SEQUENCE_ID_LEN . . . . .	1015
8.32.2.15 VTSS_PHY_TS_SIG_DEST_IP_LEN . . . . .	1016
8.32.2.16 VTSS_PHY_TS_SIG_SRC_IP_LEN . . . . .	1016
8.32.2.17 VTSS_PHY_TS_SIG_DEST_MAC_LEN . . . . .	1016
8.32.2.18 VTSS_PTP_IP_1588_VERSION_2_1 . . . . .	1016
8.32.2.19 VTSS_PHY_TS_ETH_ADDR_MATCH_48BIT . . . . .	1016
8.32.2.20 VTSS_PHY_TS_ETH_ADDR_MATCH_ANY_UNICAST . . . . .	1017
8.32.2.21 VTSS_PHY_TS_ETH_ADDR_MATCH_ANY_MULTICAST . . . . .	1017
8.32.2.22 VTSS_PHY_TS_ETH_MATCH_DEST_ADDR . . . . .	1017
8.32.2.23 VTSS_PHY_TS_ETH_MATCH_SRC_ADDR . . . . .	1017
8.32.2.24 VTSS_PHY_TS_ETH_MATCH_SRC_OR_DEST . . . . .	1017
8.32.2.25 VTSS_PHY_TS_TAG_TYPE_C . . . . .	1018
8.32.2.26 VTSS_PHY_TS_TAG_TYPE_S . . . . .	1018
8.32.2.27 VTSS_PHY_TS_TAG_TYPE_I . . . . .	1018
8.32.2.28 VTSS_PHY_TS_TAG_TYPE_B . . . . .	1018
8.32.2.29 VTSS_PHY_TS_TAG_RANGE_NONE . . . . .	1018
8.32.2.30 VTSS_PHY_TS_TAG_RANGE_OUTER . . . . .	1019
8.32.2.31 VTSS_PHY_TS_TAG_RANGE_INNER . . . . .	1019
8.32.2.32 VTSS_PHY_TS_IP_VER_4 . . . . .	1019
8.32.2.33 VTSS_PHY_TS_IP_VER_6 . . . . .	1019
8.32.2.34 VTSS_PHY_TS_IP_MATCH_SRC . . . . .	1019
8.32.2.35 VTSS_PHY_TS_IP_MATCH_DEST . . . . .	1020
8.32.2.36 VTSS_PHY_TS_IP_MATCH_SRC_OR_DEST . . . . .	1020
8.32.2.37 VTSS_PHY_TS_MPLS_STACK_DEPTH_1 . . . . .	1020
8.32.2.38 VTSS_PHY_TS_MPLS_STACK_DEPTH_2 . . . . .	1020
8.32.2.39 VTSS_PHY_TS_MPLS_STACK_DEPTH_3 . . . . .	1020
8.32.2.40 VTSS_PHY_TS_MPLS_STACK_DEPTH_4 . . . . .	1021

8.32.2.41 VTSS_PHY_TS_MPLS_STACK_REF_POINT_TOP . . . . .	1021
8.32.2.42 VTSS_PHY_TS_MPLS_STACK_REF_POINT_END . . . . .	1021
8.32.2.43 VTSS_PHY_TS_ENG_FLOW_VALID_FOR_CH0 . . . . .	1021
8.32.2.44 VTSS_PHY_TS_ENG_FLOW_VALID_FOR_CH1 . . . . .	1021
8.32.2.45 VTSS_PHY_TS_INGR_ENGINE_ERR . . . . .	1022
8.32.2.46 VTSS_PHY_TS_INGR_RW_PREAM_ERR . . . . .	1022
8.32.2.47 VTSS_PHY_TS_INGR_RW_FCS_ERR . . . . .	1022
8.32.2.48 VTSS_PHY_TS_EGR_ENGINE_ERR . . . . .	1022
8.32.2.49 VTSS_PHY_TS_EGR_RW_FCS_ERR . . . . .	1022
8.32.2.50 VTSS_PHY_TS_EGR_TIMESTAMP_CAPTURED . . . . .	1023
8.32.2.51 VTSS_PHY_TS_EGR_FIFO_OVERFLOW . . . . .	1023
8.32.2.52 VTSS_PHY_TS_DATA_IN_RSRVD_FIELD . . . . .	1023
8.32.2.53 VTSS_PHY_TS_LTC_NEW_PPS_INTRPT . . . . .	1023
8.32.2.54 VTSS_PHY_TS_LTC_LOAD_SAVE_NEW_TOD . . . . .	1023
8.32.2.55 VTSS_PHY_TS_8487_XAUI_SEL_0 . . . . .	1024
8.32.2.56 VTSS_PHY_TS_8487_XAUI_SEL_1 . . . . .	1024
8.32.2.57 VTSS_PHY_TS_INGR_DATAPATH_RESET . . . . .	1024
8.32.2.58 VTSS_PHY_TS_EGR_DATAPATH_RESET . . . . .	1024
8.32.2.59 VTSS_PHY_TS_INGR_LTC1_RESET . . . . .	1024
8.32.2.60 VTSS_PHY_TS_EGR_LTC2_RESET . . . . .	1025
8.32.2.61 VTSS_PHY_TS_EGR_FIFO_RESET . . . . .	1025
8.32.3 Typedef Documentation . . . . .	1025
8.32.3.1 vtss_phy_ts_alt_clock_mode_t . . . . .	1025
8.32.3.2 vtss_phy_ts_scaled_ppb_t . . . . .	1025
8.32.3.3 vtss_phy_ts_fifo_sig_mask_t . . . . .	1026
8.32.3.4 vtss_phy_ts_fifo_read . . . . .	1026
8.32.3.5 vtss_phy_ts_engine_channel_map_t . . . . .	1026
8.32.3.6 vtss_phy_ts_event_t . . . . .	1026
8.32.3.7 vtss_phy_ts_8487_xaui_sel_t . . . . .	1026
8.32.3.8 vtss_phy_ts_soft_reset_t . . . . .	1027

8.32.4 Enumeration Type Documentation . . . . .	1027
8.32.4.1 <code>vtss_phy_ts_todadj_status_t</code> . . . . .	1027
8.32.4.2 <code>vtss_phy_ts_fifo_status_t</code> . . . . .	1027
8.32.4.3 <code>vtss_phy_ts_encap_t</code> . . . . .	1028
8.32.4.4 <code>vtss_phy_ts_engine_t</code> . . . . .	1028
8.32.4.5 <code>vtss_phy_ts_engine_flow_match_t</code> . . . . .	1028
8.32.4.6 <code>vtss_phy_ts_ptp_clock_mode_t</code> . . . . .	1029
8.32.4.7 <code>vtss_phy_ts_ptp_delaym_type_t</code> . . . . .	1029
8.32.4.8 <code>vtss_phy_ts_y1731_oam_delaym_type_t</code> . . . . .	1030
8.32.4.9 <code>vtss_phy_ts_ietf_mpls_ach_oam_delaym_type_t</code> . . . . .	1030
8.32.4.10 <code>vtss_phy_ts_ietf_mpls_ach_oam_ts_format_t</code> . . . . .	1031
8.32.4.11 <code>vtss_phy_ts_action_format</code> . . . . .	1031
8.32.4.12 <code>vtss_phy_ts_clockfreq_t</code> . . . . .	1031
8.32.4.13 <code>vtss_phy_ts_clock_src_t</code> . . . . .	1032
8.32.4.14 <code>vtss_phy_ts_rxtimestamp_pos_t</code> . . . . .	1032
8.32.4.15 <code>vtss_phy_ts_rxtimestamp_len_t</code> . . . . .	1033
8.32.4.16 <code>vtss_phy_ts_fifo_mode_t</code> . . . . .	1033
8.32.4.17 <code>vtss_phy_ts_fifo_timestamp_len_t</code> . . . . .	1033
8.32.4.18 <code>vtss_phy_ts_tc_op_mode_t</code> . . . . .	1034
8.32.4.19 <code>vtss_phy_ts_nphase_sampler_t</code> . . . . .	1034
8.32.4.20 <code>vtss_phy_ts_ptp_message_type_t</code> . . . . .	1035
8.32.5 Function Documentation . . . . .	1035
8.32.5.1 <code>vtss_phy_ts_alt_clock_saved_get()</code> . . . . .	1035
8.32.5.2 <code>vtss_phy_ts_alt_clock_mode_get()</code> . . . . .	1035
8.32.5.3 <code>vtss_phy_ts_alt_clock_mode_set()</code> . . . . .	1036
8.32.5.4 <code>vtss_phy_ts_pps_conf_set()</code> . . . . .	1036
8.32.5.5 <code>vtss_phy_ts_pps_conf_get()</code> . . . . .	1037
8.32.5.6 <code>vtss_phy_ts_ingress_latency_set()</code> . . . . .	1037
8.32.5.7 <code>vtss_phy_ts_ingress_latency_get()</code> . . . . .	1038
8.32.5.8 <code>vtss_phy_ts_egress_latency_set()</code> . . . . .	1038

8.32.5.9 vtss_phy_ts_egress_latency_get()	1039
8.32.5.10 vtss_phy_ts_path_delay_set()	1039
8.32.5.11 vtss_phy_ts_path_delay_get()	1040
8.32.5.12 vtss_phy_ts_delay_asymmetry_set()	1040
8.32.5.13 vtss_phy_ts_delay_asymmetry_get()	1041
8.32.5.14 vtss_phy_ts_ptptime_set()	1041
8.32.5.15 vtss_phy_ts_ptptime_set_done()	1042
8.32.5.16 vtss_phy_ts_ptptime_arm()	1042
8.32.5.17 vtss_phy_ts_ptptime_get()	1043
8.32.5.18 vtss_phy_ts_load_ptptime_get()	1043
8.32.5.19 vtss_phy_ts_sertod_set()	1043
8.32.5.20 vtss_phy_ts_sertod_get()	1044
8.32.5.21 vtss_phy_ts_loadpulse_delay_set()	1044
8.32.5.22 vtss_phy_ts_loadpulse_delay_get()	1045
8.32.5.23 vtss_phy_ts_clock_rateadj_set()	1045
8.32.5.24 vtss_phy_ts_clock_rateadj_get()	1046
8.32.5.25 vtss_phy_ts_clock_rateadj_ppm_set()	1046
8.32.5.26 vtss_phy_ts_clock_rateadj_ppm_get()	1046
8.32.5.27 vtss_phy_ts_ptptime_adj1ns()	1047
8.32.5.28 vtss_phy_ts_timeofday_offset_set()	1047
8.32.5.29 vtss_phy_ts_ongoing_adjustment()	1048
8.32.5.30 vtss_phy_ts_ltc_freq_synth_pulse_set()	1048
8.32.5.31 vtss_phy_ts_ltc_freq_synth_pulse_get()	1049
8.32.5.32 vtss_phy_daisy_conf_set()	1049
8.32.5.33 vtss_phy_daisy_conf_get()	1049
8.32.5.34 vtss_phy_ts_fifo_sig_set()	1050
8.32.5.35 vtss_phy_ts_fifo_sig_get()	1050
8.32.5.36 vtss_phy_ts_fifo_empty()	1051
8.32.5.37 vtss_phy_ts_fifo_read_install()	1051
8.32.5.38 vtss_phy_ts_fifo_read_cb_get()	1052

8.32.5.39 vtss_phy_ts_ingress_engine_init()	1052
8.32.5.40 vtss_phy_ts_ingress_engine_init_conf_get()	1053
8.32.5.41 vtss_phy_ts_ingress_engine_clear()	1053
8.32.5.42 vtss_phy_ts_egress_engine_init()	1054
8.32.5.43 vtss_phy_ts_egress_engine_init_conf_get()	1055
8.32.5.44 vtss_phy_ts_egress_engine_clear()	1055
8.32.5.45 vtss_phy_ts_ingress_engine_conf_set()	1056
8.32.5.46 vtss_phy_ts_ingress_engine_conf_get()	1056
8.32.5.47 vtss_phy_ts_egress_engine_conf_set()	1057
8.32.5.48 vtss_phy_ts_egress_engine_conf_get()	1057
8.32.5.49 vtss_phy_ts_ingress_engine_action_set()	1058
8.32.5.50 vtss_phy_ts_ingress_engine_action_get()	1058
8.32.5.51 vtss_phy_ts_egress_engine_action_set()	1060
8.32.5.52 vtss_phy_ts_egress_engine_action_get()	1060
8.32.5.53 vtss_phy_ts_event_enable_set()	1062
8.32.5.54 vtss_phy_ts_event_enable_get()	1062
8.32.5.55 vtss_phy_ts_event_poll()	1063
8.32.5.56 vtss_phy_ts_stats_get()	1063
8.32.5.57 vtss_phy_ts_correction_overflow_get()	1064
8.32.5.58 vtss_phy_ts_mode_set()	1064
8.32.5.59 vtss_phy_ts_mode_get()	1065
8.32.5.60 vtss_phy_ts_init()	1065
8.32.5.61 vtss_phy_ts_init_conf_get()	1065
8.32.5.62 vtss_phy_ts_nphase_status_get()	1066
8.32.5.63 vtss_phy_ts_hiacc_set()	1066
8.32.5.64 vtss_phy_ts_hiacc_get()	1067
8.32.5.65 vtss_phy_ts_block_soft_reset()	1067
8.32.5.66 vtss_phy_ts_new_spi_mode_set()	1068
8.32.5.67 vtss_phy_ts_new_spi_mode_get()	1068
8.32.5.68 vtss_phy_ts_phy_oper_mode_change()	1069

8.32.5.69 <code>vtss_phy_1588_csr_reg_write()</code>	1069
8.32.5.70 <code>vtss_phy_1588_csr_reg_read()</code>	1070
8.32.5.71 <code>vtss_phy_ts_status_check()</code>	1070
8.32.5.72 <code>vtss_phy_ts_10g_extended_fifo_sync()</code>	1071
8.32.5.73 <code>vtss_phy_ts_10g_fifo_sync()</code>	1071
8.32.5.74 <code>vtss_phy_ts_bypass_clear()</code>	1072
8.32.5.75 <code>vtss_phy_ts_viper_fifo_reset()</code>	1072
8.32.5.76 <code>vtss_phy_ts_tesla_tsp_fifo_sync()</code>	1072
8.32.5.77 <code>vtss_phy_1g_ts_fifo_sync()</code>	1073
8.32.5.78 <code>vtss_phy_1588_debug_reg_read()</code>	1074
8.32.5.79 <code>vtss_phy_ts_flow_clear_cf_set()</code>	1074
8.33 <code>vtss_api/include/vtss_port_api.h</code> File Reference	1075
8.33.1 Detailed Description	1077
8.33.2 Macro Definition Documentation	1077
8.33.2.1 <code>CHIP_PORT_UNUSED</code>	1077
8.33.2.2 <code>VTSS_FRAME_GAP_DEFAULT</code>	1078
8.33.2.3 <code>VTSS_MAX_FRAME_LENGTH_STANDARD</code>	1078
8.33.2.4 <code>VTSS_MAX_FRAME_LENGTH_MAX [1/2]</code>	1078
8.33.2.5 <code>VTSS_MAX_FRAME_LENGTH_MAX [2/2]</code>	1078
8.33.3 Enumeration Type Documentation	1078
8.33.3.1 <code>vtss_miim_controller_t</code>	1078
8.33.3.2 <code>vtss_internal_bw_t</code>	1079
8.33.3.3 <code>vtss_port_clause_37_remote_fault_t</code>	1079
8.33.3.4 <code>vtss_port_max_tags_t</code>	1080
8.33.3.5 <code>vtss_port_loop_t</code>	1080
8.33.3.6 <code>vtss_port_forward_t</code>	1080
8.33.4 Function Documentation	1081
8.33.4.1 <code>vtss_port_map_set()</code>	1081
8.33.4.2 <code>vtss_port_map_get()</code>	1081
8.33.4.3 <code>vtss_port_clause_37_control_get()</code>	1081

8.33.4.4	vtss_port_clause_37_control_set()	1082
8.33.4.5	vtss_port_conf_set()	1082
8.33.4.6	vtss_port_conf_get()	1083
8.33.4.7	vtss_port_status_get()	1083
8.33.4.8	vtss_port_counters_update()	1084
8.33.4.9	vtss_port_counters_clear()	1084
8.33.4.10	vtss_port_counters_get()	1084
8.33.4.11	vtss_port_basic_counters_get()	1085
8.33.4.12	vtss_port_forward_state_get()	1085
8.33.4.13	vtss_port_forward_state_set()	1086
8.33.4.14	vtss_port_ifh_conf_set()	1086
8.33.4.15	vtss_port_ifh_conf_get()	1087
8.33.4.16	vtss_miim_read()	1087
8.33.4.17	vtss_miim_write()	1088
8.33.4.18	vtss_port_mmd_read()	1088
8.33.4.19	vtss_port_mmd_read_inc()	1089
8.33.4.20	vtss_port_mmd_write()	1089
8.33.4.21	vtss_port_mmd_masked_write()	1090
8.33.4.22	vtss_mmd_read()	1090
8.33.4.23	vtss_mmd_write()	1091
8.34	vtss_api/include/vtss_qos_api.h File Reference	1091
8.34.1	Detailed Description	1093
8.34.2	Macro Definition Documentation	1093
8.34.2.1	VTSS_PORT_POLICERS	1094
8.34.2.2	VTSS_PORT_POLICER_CPU_QUEUES	1094
8.34.2.3	VTSS_QCL_IDS	1094
8.34.2.4	VTSS_QCL_ID_START	1094
8.34.2.5	VTSS_QCL_ID_END	1094
8.34.2.6	VTSS_QCL_ARRAY_SIZE	1095
8.34.2.7	VTSS_QCE_ID_LAST	1095

8.34.3 Enumeration Type Documentation . . . . .	1095
8.34.3.1 <code>vtss_tag_remark_mode_t</code> . . . . .	1095
8.34.3.2 <code>vtss_dscp_mode_t</code> . . . . .	1095
8.34.3.3 <code>vtss_dscp_emode_t</code> . . . . .	1096
8.34.3.4 <code>vtss_qce_type_t</code> . . . . .	1096
8.34.4 Function Documentation . . . . .	1096
8.34.4.1 <code>vtss_qos_conf_get()</code> . . . . .	1096
8.34.4.2 <code>vtss_qos_conf_set()</code> . . . . .	1097
8.34.4.3 <code>vtss_qos_port_conf_get()</code> . . . . .	1097
8.34.4.4 <code>vtss_qos_port_conf_set()</code> . . . . .	1098
8.34.4.5 <code>vtss_qce_init()</code> . . . . .	1098
8.34.4.6 <code>vtss_qce_add()</code> . . . . .	1098
8.34.4.7 <code>vtss_qce_del()</code> . . . . .	1099
8.35 <code>vtss_api/include/vtss_rab_api.h</code> File Reference . . . . .	1099
8.35.1 Detailed Description . . . . .	1099
8.36 <code>vtss_api/include/vtss_security_api.h</code> File Reference . . . . .	1100
8.36.1 Detailed Description . . . . .	1102
8.36.2 Macro Definition Documentation . . . . .	1102
8.36.2.1 <code>VTSS_ACE_ID_LAST</code> . . . . .	1102
8.36.2.2 <code>VTSS_PORT_NO_ANY</code> . . . . .	1102
8.36.3 Enumeration Type Documentation . . . . .	1102
8.36.3.1 <code>vtss_auth_state_t</code> . . . . .	1102
8.36.3.2 <code>vtss_ace_type_t</code> . . . . .	1103
8.36.3.3 <code>vtss_ace_bit_t</code> . . . . .	1103
8.36.4 Function Documentation . . . . .	1103
8.36.4.1 <code>vtss_auth_port_state_get()</code> . . . . .	1104
8.36.4.2 <code>vtss_auth_port_state_set()</code> . . . . .	1104
8.36.4.3 <code>vtss_acl_policer_conf_get()</code> . . . . .	1104
8.36.4.4 <code>vtss_acl_policer_conf_set()</code> . . . . .	1105
8.36.4.5 <code>vtss_acl_port_conf_get()</code> . . . . .	1105

8.36.4.6 vtss_acl_port_conf_set() . . . . .	1106
8.36.4.7 vtss_acl_port_counter_get() . . . . .	1106
8.36.4.8 vtss_acl_port_counter_clear() . . . . .	1107
8.36.4.9 vtss_ace_init() . . . . .	1107
8.36.4.10 vtss_ace_add() . . . . .	1107
8.36.4.11 vtss_ace_del() . . . . .	1108
8.36.4.12 vtss_ace_counter_get() . . . . .	1108
8.36.4.13 vtss_ace_counter_clear() . . . . .	1109
8.37 vtss_api/include/vtss_sfi4_api.h File Reference . . . . .	1109
8.37.1 Detailed Description . . . . .	1109
8.38 vtss_api/include/vtss_sync_api.h File Reference . . . . .	1109
8.38.1 Detailed Description . . . . .	1110
8.38.2 Macro Definition Documentation . . . . .	1110
8.38.2.1 VTSS_SYNCE_CLK_A . . . . .	1110
8.38.2.2 VTSS_SYNCE_CLK_B . . . . .	1111
8.38.2.3 VTSS_SYNCE_CLK_MAX . . . . .	1111
8.38.3 Enumeration Type Documentation . . . . .	1111
8.38.3.1 vtss_syncce_divider_t . . . . .	1111
8.38.4 Function Documentation . . . . .	1111
8.38.4.1 vtss_syncce_clock_out_set() . . . . .	1111
8.38.4.2 vtss_syncce_clock_out_get() . . . . .	1112
8.38.4.3 vtss_syncce_clock_in_set() . . . . .	1112
8.38.4.4 vtss_syncce_clock_in_get() . . . . .	1113
8.39 vtss_api/include/vtss_tfi5_api.h File Reference . . . . .	1113
8.39.1 Detailed Description . . . . .	1113
8.40 vtss_api/include/vtss_ts_api.h File Reference . . . . .	1113
8.40.1 Detailed Description . . . . .	1116
8.40.2 Function Documentation . . . . .	1117
8.40.2.1 vtss_ts_timeofday_set() . . . . .	1117
8.40.2.2 vtss_ts_timeofday_set_delta() . . . . .	1117

---

8.40.2.3 vtss_ts_timeofday_offset_set()	1117
8.40.2.4 vtss_ts_adjtimer_one_sec()	1119
8.40.2.5 vtss_ts_ongoing_adjustment()	1119
8.40.2.6 vtss_ts_timeofday_get()	1120
8.40.2.7 vtss_ts_timeofday_next_pps_get()	1120
8.40.2.8 vtss_ts_adjtimer_set()	1120
8.40.2.9 vtss_ts_adjtimer_get()	1121
8.40.2.10 vtss_ts_freq_offset_get()	1121
8.40.2.11 vtss_ts_external_clock_mode_get()	1122
8.40.2.12 vtss_ts_external_clock_mode_set()	1122
8.40.2.13 vtss_ts_external_clock_saved_get()	1122
8.40.2.14 vtss_ts_ingress_latency_set()	1123
8.40.2.15 vtss_ts_ingress_latency_get()	1123
8.40.2.16 vtss_ts_p2p_delay_set()	1124
8.40.2.17 vtss_ts_p2p_delay_get()	1124
8.40.2.18 vtss_ts_egress_latency_set()	1125
8.40.2.19 vtss_ts_egress_latency_get()	1125
8.40.2.20 vtss_ts_delay_asymmetry_set()	1125
8.40.2.21 vtss_ts_delay_asymmetry_get()	1126
8.40.2.22 vtss_ts_operation_mode_set()	1126
8.40.2.23 vtss_ts_operation_mode_get()	1127
8.40.2.24 vtss_ts_internal_mode_set()	1127
8.40.2.25 vtss_ts_internal_mode_get()	1127
8.40.2.26 vtss_tx_timestamp_update()	1128
8.40.2.27 vtss_rx_timestamp_get()	1128
8.40.2.28 vtss_rx_timestamp_id_release()	1129
8.40.2.29 vtss_rx_master_timestamp_get()	1129
8.40.2.30 vtss_tx_timestamp_idx_alloc()	1130
8.40.2.31 vtss_timestamp_age()	1130
8.40.2.32 vtss_ts_status_change()	1130

8.41 vtss_api/include/vtss_upi_api.h File Reference . . . . .	1131
8.41.1 Detailed Description . . . . .	1131
8.42 vtss_api/include/vtss_wis_api.h File Reference . . . . .	1131
8.42.1 Detailed Description . . . . .	1136
8.42.2 Macro Definition Documentation . . . . .	1136
8.42.2.1 VTSS_EWIS_SEF_EV . . . . .	1137
8.42.2.2 VTSS_EWIS_FPLM_EV . . . . .	1137
8.42.2.3 VTSS_EWIS_FAIS_EV . . . . .	1137
8.42.2.4 VTSS_EWIS_LOF_EV . . . . .	1137
8.42.2.5 VTSS_EWIS_LOS_EV . . . . .	1137
8.42.2.6 VTSS_EWIS_RDIL_EV . . . . .	1138
8.42.2.7 VTSS_EWIS_AISL_EV . . . . .	1138
8.42.2.8 VTSS_EWIS_LCDP_EV . . . . .	1138
8.42.2.9 VTSS_EWIS_PLMP_EV . . . . .	1138
8.42.2.10 VTSS_EWIS_AISP_EV . . . . .	1138
8.42.2.11 VTSS_EWIS_LOPP_EV . . . . .	1139
8.42.2.12 VTSS_EWIS_MODULE_EV . . . . .	1139
8.42.2.13 VTSS_EWIS_TXLOL_EV . . . . .	1139
8.42.2.14 VTSS_EWIS_RXLOL_EV . . . . .	1139
8.42.2.15 VTSS_EWIS_LOPC_EV . . . . .	1139
8.42.2.16 VTSS_EWIS_UNEQP_EV . . . . .	1140
8.42.2.17 VTSS_EWIS_FEUNEQP_EV . . . . .	1140
8.42.2.18 VTSS_EWIS_FERDIP_EV . . . . .	1140
8.42.2.19 VTSS_EWIS_REIL_EV . . . . .	1140
8.42.2.20 VTSS_EWIS_REIP_EV . . . . .	1140
8.42.2.21 VTSS_EWIS_HIGH_BER_EV . . . . .	1141
8.42.2.22 VTSS_EWIS_PCS_RECEIVE_FAULT_PEND . . . . .	1141
8.42.2.23 VTSS_EWIS_B1_NZ_EV . . . . .	1141
8.42.2.24 VTSS_EWIS_B2_NZ_EV . . . . .	1141
8.42.2.25 VTSS_EWIS_B3_NZ_EV . . . . .	1141

8.42.2.26 VTSS_EWIS_REIL_NZ_EV . . . . .	1142
8.42.2.27 VTSS_EWIS_REIP_NZ_EV . . . . .	1142
8.42.2.28 VTSS_EWIS_B1_THRESH_EV . . . . .	1142
8.42.2.29 VTSS_EWIS_B2_THRESH_EV . . . . .	1142
8.42.2.30 VTSS_EWIS_B3_THRESH_EV . . . . .	1142
8.42.2.31 VTSS_EWIS_REIL_THRESH_EV . . . . .	1143
8.42.2.32 VTSS_EWIS_REIP_THRESH_EV . . . . .	1143
8.42.3 Typedef Documentation . . . . .	1143
8.42.3.1 vtss_ewis_static_conf_t . . . . .	1143
8.42.3.2 vtss_ewis_event_t . . . . .	1143
8.42.4 Enumeration Type Documentation . . . . .	1143
8.42.4.1 vtss_ewis_tti_mode_t . . . . .	1143
8.42.4.2 vtss_ewis_perf_cntr_mode_t . . . . .	1144
8.42.4.3 vtss_ewis_mode_t . . . . .	1144
8.42.4.4 vtss_ewis_test_pattern_s . . . . .	1144
8.42.4.5 vtss_ewis_prbs31_err_inj_t . . . . .	1145
8.42.5 Function Documentation . . . . .	1145
8.42.5.1 vtss_ewis_event_enable() . . . . .	1145
8.42.5.2 vtss_ewis_event_poll() . . . . .	1146
8.42.5.3 vtss_ewis_event_poll_without_mask() . . . . .	1146
8.42.5.4 vtss_ewis_event_force() . . . . .	1147
8.42.5.5 vtss_ewis_static_conf_get() . . . . .	1147
8.42.5.6 vtss_ewis_force_conf_set() . . . . .	1148
8.42.5.7 vtss_ewis_force_conf_get() . . . . .	1148
8.42.5.8 vtss_ewis_tx_oh_set() . . . . .	1149
8.42.5.9 vtss_ewis_tx_oh_get() . . . . .	1149
8.42.5.10 vtss_ewis_tx_oh_passthru_set() . . . . .	1150
8.42.5.11 vtss_ewis_tx_oh_passthru_get() . . . . .	1150
8.42.5.12 vtss_ewis_mode_set() . . . . .	1150
8.42.5.13 vtss_ewis_mode_get() . . . . .	1151

8.42.5.14 vtss_ewis_reset()	1151
8.42.5.15 vtss_ewis_cons_act_set()	1152
8.42.5.16 vtss_ewis_cons_act_get()	1152
8.42.5.17 vtss_ewis_section_txti_set()	1153
8.42.5.18 vtss_ewis_section_txti_get()	1153
8.42.5.19 vtss_ewis_exp_sl_set()	1154
8.42.5.20 vtss_ewis_path_txti_set()	1154
8.42.5.21 vtss_ewis_path_txti_get()	1154
8.42.5.22 vtss_ewis_test_mode_set()	1156
8.42.5.23 vtss_ewis_test_mode_get()	1156
8.42.5.24 vtss_ewis_prbs31_err_inj_set()	1157
8.42.5.25 vtss_ewis_test_counter_get()	1157
8.42.5.26 vtss_ewis_defects_get()	1158
8.42.5.27 vtss_ewis_status_get()	1158
8.42.5.28 vtss_ewis_section_acti_get()	1159
8.42.5.29 vtss_ewis_path_acti_get()	1159
8.42.5.30 vtss_ewis_counter_get()	1160
8.42.5.31 vtss_ewis_perf_get()	1160
8.42.5.32 vtss_ewis_counter_threshold_set()	1161
8.42.5.33 vtss_ewis_counter_threshold_get()	1161
8.42.5.34 vtss_ewis_perf_mode_set()	1161
8.42.5.35 vtss_ewis_perf_mode_get()	1163
8.43 vtss_api/include/vtss_xaui_api.h File Reference	1163
8.43.1 Detailed Description	1163
8.44 vtss_api/include/vtss_xfi_api.h File Reference	1163
8.44.1 Detailed Description	1164
Index	1165

# 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 identified 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:

<a href="#">ib_par_cfg</a>	Generalized data structure for IB parameters . . . . .	31
<a href="#">port_custom_conf_t</a>	Port configuration . . . . .	32
<a href="#">serdes_fields_t</a>	Serdes fields . . . . .	35
<a href="#">tag_vtss_fdma_list</a>	Software DCB structure . . . . .	36
<a href="#">vtss_ace_frame_arp_t</a>	Frame data for VTSS_ACE_TYPE_ARP . . . . .	40
<a href="#">vtss_ace_frame_etype_t</a>	Frame data for VTSS_ACE_TYPEETYPE . . . . .	42
<a href="#">vtss_ace_frame_ipv4_t</a>	Frame data for VTSS_ACE_TYPE_IPV4 . . . . .	44
<a href="#">vtss_ace_frame_ipv6_t</a>	Frame data for VTSS_ACE_TYPE_IPV6 . . . . .	48
<a href="#">vtss_ace_frame_llc_t</a>	Frame data for VTSS_ACE_TYPE_LLCC . . . . .	52
<a href="#">vtss_ace_frame_snap_t</a>	Frame data for VTSS_ACE_TYPE_SNAP . . . . .	54
<a href="#">vtss_ace_t</a>	Access Control Entry . . . . .	55
<a href="#">vtss_ace_vlan_t</a>	ACE VLAN information . . . . .	59
<a href="#">vtss_acl_action_t</a>	ACL Action . . . . .	60
<a href="#">vtss_acl_policer_conf_t</a>	ACL policer configuration . . . . .	62
<a href="#">vtss_acl_port_conf_t</a>	ACL port configuration . . . . .	63
<a href="#">vtss_aggr_mode_t</a>	Aggregation traffic distribution mode . . . . .	64
<a href="#">vtss_aneg_t</a>	Auto negotiation struct . . . . .	66
<a href="#">vtss_api_lock_t</a>	API lock structure . . . . .	66

vtss_basic_counters_t	Basic counters structure . . . . .	68
vtss_chip_id_t	Chip ID . . . . .	69
vtss_counter_pair_t	Counter pair . . . . .	70
vtss_debug_info_t	Debug information structure . . . . .	71
vtss_debug_lock_t	API debug lock structure . . . . .	73
vtss_dgroup_port_conf_t	Destination group port configuration . . . . .	74
vtss_dlb_policer_conf_t	Dual leaky buckets policer configuration . . . . .	74
vtss_ece_action_t	ECE action . . . . .	77
vtss_ece_frame_ipv4_t	ECE IPv4 information . . . . .	79
vtss_ece_frame_ipv6_t	ECE IPv6 information . . . . .	79
vtss_ece_inner_tag_t	ECE inner tag . . . . .	80
vtss_ece_key_t	ECE key . . . . .	82
vtss_ece_mac_t	ECE MAC information . . . . .	84
vtss_ece_outer_tag_t	ECE outer tag . . . . .	85
vtss_ece_t	EVC Control Entry . . . . .	86
vtss_ece_tag_t	ECE tag information . . . . .	87
vtss_eee_port_conf_t	EEE port configuration . . . . .	89
vtss_eee_port_counter_t	EEE port counters (JR only) . . . . .	90
vtss_eee_port_state_t	EEE port state (JR only) . . . . .	92
vtss_eps_port_conf_t	Port protection configuration . . . . .	93
vtss_evc_conf_t	EVC configuration (excluding UNIs) . . . . .	94
vtss_evc_counters_t	EVC/ECE counters . . . . .	95
vtss_evc_pb_conf_t	PB specific EVC configuration . . . . .	97
vtss_evc_port_conf_t	EVC port configuration . . . . .	99
vtss_ewis_aisl_cons_act_s	EWIS AIS-L consequent actions . . . . .	100
vtss_ewis_conf_s	EWIS configuration primitives, used to hold all the configuration parameters in the internal state in the API . . . . .	101
vtss_ewis_cons_act_s	EWIS consequent actions . . . . .	104
vtss_ewis_counter_s	EWIS performance counters. These counters are free running counters that wraps to zero . . . . .	105

<b>vtss_ewis_counter_threshold_s</b>	EWIS performance counter thresholds . . . . .	107
<b>vtss_ewis_defects_s</b>	EWIS defects . . . . .	108
<b>vtss_ewis_fault_cons_act_s</b>	EWIS fault mask configuration, i.e set up which defects trigger the Fault condition . . . . .	112
<b>vtss_ewis_force_mode_s</b>	EWIS force modes . . . . .	115
<b>vtss_ewis_line_force_mode_s</b>	EWIS line force mode . . . . .	116
<b>vtss_ewis_line_tx_force_mode_s</b>	EWIS line TX force mode . . . . .	117
<b>vtss_ewis_path_force_mode_s</b>	EWIS path force modes . . . . .	118
<b>vtss_ewis_perf_mode_s</b>	EWIS Mode(Bit/Block) for the Performance Monitoring Counters . . . . .	119
<b>vtss_ewis_perf_s</b>	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 . . . . .	121
<b>vtss_ewis_rdil_cons_act_s</b>	EWIS RDI-L consequent actions . . . . .	122
<b>vtss_ewis_sl_conf_s</b>	Signal label configuration . . . . .	124
<b>vtss_ewis_static_conf_s</b>	EWIS static configuration data, . . . . .	124
<b>vtss_ewis_status_s</b>	EWIS status . . . . .	128
<b>vtss_ewis_test_conf_s</b>	EWIS test configuration . . . . .	129
<b>vtss_ewis_test_status_s</b>	EWIS test status . . . . .	130
<b>vtss_ewis_tti_s</b>	Trail Trace Identifier type . . . . .	131
<b>vtss_ewis_tx_oh_s</b>	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 . . . . .	132
<b>vtss_ewis_tx_passthru_s</b>	EWIS overhead passthru configuration . . . . .	136
<b>vtss_fan_conf_t</b>	Fan specifications . . . . .	140
<b>vtss_fdma_cfg_t</b>	FDMA configuration structure . . . . .	141
<b>vtss_fdma_ch_cfg_t</b>	Channel configuration structure . . . . .	147
<b>vtss_fdma_throttle_cfg_t</b>	. . . . .	151
<b>vtss_fdma_tx_info_t</b>	FDMA Injection Properties . . . . .	152
<b>vtss_gpio_10g_gpio_mode_t</b>	GPIO configured mode . . . . .	156
<b>vtss_gpio_10g_led_conf_t</b>	LED Mode selection . . . . .	159
<b>vtss_init_conf_t</b>	Initialization configuration . . . . .	159
<b>vtss_inst_create_t</b>	Create structure . . . . .	164

<a href="#">vtss_ip_addr_t</a>	Either an IPv4 or IPv6 address . . . . .	164
<a href="#">vtss_ip_network_t</a>	IPv6 network . . . . .	166
<a href="#">vtss_ipv4_network_t</a>	IPv4 network . . . . .	166
<a href="#">vtss_ipv4_uc_t</a>	IPv4 unicast routing entry . . . . .	167
<a href="#">vtss_ipv6_network_t</a>	IPv6 network . . . . .	168
<a href="#">vtss_ipv6_t</a>	IPv6 address/mask . . . . .	169
<a href="#">vtss_ipv6_uc_t</a>	IPv6 routing entry . . . . .	170
<a href="#">vtss_irq_conf_t</a>	Interrupt configuration options . . . . .	171
<a href="#">vtss_irq_status_t</a>	Interrupt status structure . . . . .	172
<a href="#">vtss_l3_common_conf_t</a>	Common configurations for all routing legs . . . . .	173
<a href="#">vtss_l3_counters_t</a>	Routing interface statics counter . . . . .	174
<a href="#">vtss_l3_neighbour_t</a>	Neighbour entry . . . . .	176
<a href="#">vtss_l3_rleg_conf_t</a>	Router leg control structure . . . . .	178
<a href="#">vtss_lcpll_status_t</a>	Structure for Get PHY LC-PLL status . . . . .	180
<a href="#">vtss_learn_mode_t</a>	Learning mode . . . . .	182
<a href="#">vtss_mac_t</a>	MAC Address . . . . .	183
<a href="#">vtss_mac_table_entry_t</a>	MAC address entry . . . . .	184
<a href="#">vtss_mac_table_status_t</a>	MAC address table status . . . . .	187
<a href="#">vtss_mirror_conf_t</a>	Mirror configuration . . . . .	188
<a href="#">vtss_mtimer_t</a>	Timer structure . . . . .	190
<a href="#">vtss_npi_conf_t</a>	NPI configuration . . . . .	191
<a href="#">vtss_os_timestamp_t</a>	. . . . .	192
<a href="#">vtss_packet_dma_conf_t</a>	. . . . .	193
<a href="#">vtss_packet_frame_info_t</a>	Information about frame . . . . .	194
<a href="#">vtss_packet_port_filter_t</a>	Packet information for each port . . . . .	195
<a href="#">vtss_packet_port_info_t</a>	Port info structure . . . . .	196
<a href="#">vtss_packet_rx_conf_t</a>	CPU Rx configuration . . . . .	198
<a href="#">vtss_packet_rx_header_t</a>	System frame header describing received frame . . . . .	199
<a href="#">vtss_packet_rx_info_t</a>	Decoded extraction header properties . . . . .	201
<a href="#">vtss_packet_rx_meta_t</a>	Input structure to <a href="#">vtss_packet_rx_hdr_decode()</a> . . . . .	211

<a href="#">vtss_packet_rx_port_conf_t</a>	Packet registration per port . . . . .	215
<a href="#">vtss_packet_rx_queue_conf_t</a>	CPU Rx queue configuration . . . . .	216
<a href="#">vtss_packet_rx_queue_map_t</a>	CPU Rx queue map . . . . .	217
<a href="#">vtss_packet_rx_queue_npi_conf_t</a>	CPU Rx queue NPI configuration . . . . .	220
<a href="#">vtss_packet_rx_reg_t</a>	CPU Rx packet registration . . . . .	221
<a href="#">vtss_packet_tx_ifh_t</a>	Compiled Tx Frame Header . . . . .	222
<a href="#">vtss_packet_tx_info_t</a>	Injection Properties . . . . .	223
<a href="#">vtss_phy_10g_apc_conf_t</a>	10G Phy APC configuration . . . . .	234
<a href="#">vtss_phy_10g_apc_status_t</a>	10G Phy APC status . . . . .	235
<a href="#">vtss_phy_10g_auto_failover_conf_t</a>	10G PHY Automatic Failover configuration . . . . .	236
<a href="#">vtss_phy_10g_base_kr_autoneg_t</a>	10G Phy Base KR Autoneg config . . . . .	239
<a href="#">vtss_phy_10g_base_kr_conf_t</a>	10G Phy 10f_base_kr_conf config data according to 802.3-2008 clause 72.7 Figure 72-11 . . . . .	240
<a href="#">vtss_phy_10g_base_kr_id_adv_abil_t</a>	10G Phy Base Link Advertisement capability config . . . . .	242
<a href="#">vtss_phy_10g_base_kr_status_t</a>	10G Phy Base KR Training & Autoneg status . . . . .	243
<a href="#">vtss_phy_10g_base_kr_train_aneg_t</a>	10G Phy Base KR Training & Autoneg config . . . . .	244
<a href="#">vtss_phy_10g_base_kr_training_t</a>	10G Phy Base KR Training config . . . . .	246
<a href="#">vtss_phy_10g_ckout_conf_t</a>	10G Phy CKOUT config data . . . . .	247
<a href="#">vtss_phy_10g_clause_37_adv_t</a>	Advertisement control data for Clause 37 aneg . . . . .	249
<a href="#">vtss_phy_10g_clause_37_cmn_status_t</a>	Clause 37 Auto-negotiation status for line and host . . . . .	251
<a href="#">vtss_phy_10g_clause_37_control_t</a>	Clause 37 control struct . . . . .	252
<a href="#">vtss_phy_10g_clause_37_status_t</a>	Clause 37 Auto-negotiation status . . . . .	254
<a href="#">vtss_phy_10g_clk_src_t</a>	10G Phy CLOCK Source Selection . . . . .	255
<a href="#">vtss_phy_10g_cnt_t</a>	10G Phy Sublayer counters . . . . .	256
<a href="#">vtss_phy_10g_fifo_sync_t</a>	10G OOS workaround options . . . . .	257
<a href="#">vtss_phy_10g_fw_status_t</a>	Firmware status . . . . .	258
<a href="#">vtss_phy_10g_host_clk_conf_t</a>	10G Phy Host clock config data . . . . .	259
<a href="#">vtss_phy_10g_i2c_slave_conf_t</a>	10G Phy I2C Master Interface for SFP Module Configuration . . . . .	260
<a href="#">vtss_phy_10g_ib_conf_t</a>	10G Phy IB configuration . . . . .	261
<a href="#">vtss_phy_10g_ib_status_t</a>	10G Phy IB configuration . . . . .	266

vtss_phy_10g_ib_storage_t	VSCOPE fast scan storage . . . . .	267
vtss_phy_10g_id_t	10G Phy part number and revision . . . . .	268
vtss_phy_10g_init_parm_t	10G Phy Initialization configuration . . . . .	270
vtss_phy_10g_jitter_conf_t	10G Phy Optimisation of jitter performance . . . . .	271
vtss_phy_10g_kr_status_aneg_t	10G Phy Base KR Autoneg status . . . . .	272
vtss_phy_10g_kr_status_fec_t	10G Phy Base KR FEC status . . . . .	274
vtss_phy_10g_kr_status_train_t	10G Phy Base KR Training status . . . . .	275
vtss_phy_10g_lane_sync_conf_t	10G Phy Lane SYNC Configuration . . . . .	277
vtss_phy_10g_line_clk_conf_t	10G Phy Line clock config data . . . . .	278
vtss_phy_10g_loopback_t	10G Phy system and network loopbacks . . . . .	279
vtss_phy_10g_mode_t	10G Phy operating mode . . . . .	280
vtss_phy_10g_ob_status_t	10G Phy OB status . . . . .	293
vtss_phy_10g_pcs_prbs_gen_conf_t	. . . . .	295
vtss_phy_10g_pcs_prbs_mon_conf_t	. . . . .	296
vtss_phy_10g_pkt_gen_conf_t	10G PHY Packet generator configuration . . . . .	297
vtss_phy_10g_pkt_mon_conf_t	10G PHY Packet Monitor configuration . . . . .	300
vtss_phy_10g_polarity_inv_t	10G Phy Polarity inversion . . . . .	302
vtss_phy_10g_prbs_gen_conf_t	. . . . .	304
vtss_phy_10g_prbs_mon_conf_t	10G PHY prbs monitor Configuration . . . . .	305
vtss_phy_10g_rxckout_conf_t	10G Phy RXCKOUT config data . . . . .	310
vtss_phy_10g_sckout_conf_t	10G Phy SCKOUT config data . . . . .	311
vtss_phy_10g_serdes_status_t	10G Phy SERDES status . . . . .	313
vtss_phy_10g_srefclk_mode_t	10G Phy srefclk config data . . . . .	320
vtss_phy_10g_status_t	10G Phy link and fault status for all sublayers . . . . .	320
vtss_phy_10g_timestamp_val_t	10G PHY timestamp value array(holder) . . . . .	323
vtss_phy_10g_txckout_conf_t	10G Phy TXCKOUT config data . . . . .	324
vtss_phy_10g_vscope_conf_t	. . . . .	325
vtss_phy_10g_vscope_scan_conf_t	VSCOPE scan configuration . . . . .	326
vtss_phy_10g_vscope_scan_status_t	. . . . .	328
vtss_phy_aneg_t	PHY auto negotiation advertisement . . . . .	330
vtss_phy_clock_conf_t	PHY clock configuration . . . . .	332

<a href="#">vtss_phy_conf_1g_t</a>	PHY 1G configuration . . . . .	333
<a href="#">vtss_phy_conf_t</a>	PHY configuration . . . . .	334
<a href="#">vtss_phy_daisy_chain_conf_t</a>	SPI daisy chain configuration . . . . .	337
<a href="#">vtss_phy_eee_conf_t</a>	EEE configuration . . . . .	338
<a href="#">vtss_phy_enhanced_led_control_t</a>	Enhanced LED control . . . . .	339
<a href="#">vtss_phy_forced_t</a>	PHY forced mode configuration . . . . .	340
<a href="#">vtss_phy_led_mode_select_t</a>	LED model selection . . . . .	341
<a href="#">vtss_phy_loopback_t</a>	1G Phy loopbacks . . . . .	342
<a href="#">vtss_phy_ltc_freq_synth_s</a>	Frequency systhesis pulse configuration . . . . .	345
<a href="#">vtss_phy_mac_serdes_pcs_cntl_t</a>	PHY MAC SerDes PCS Control, Reg16E3 . . . . .	346
<a href="#">vtss_phy_media_serdes_pcs_cntl_t</a>	PHY MEDIA SerDes PCS Control, Reg23E3 . . . . .	349
<a href="#">vtss_phy_pcs_cnt_t</a>	10G Phy PCS counters . . . . .	351
<a href="#">vtss_phy_power_conf_t</a>	PHY power configuration . . . . .	353
<a href="#">vtss_phy_power_status_t</a>	PHY power status . . . . .	354
<a href="#">vtss_phy_reset_conf_t</a>	PHY reset structure . . . . .	354
<a href="#">vtss_phy_rgmii_conf_t</a>	PHY RGMII configuration . . . . .	356
<a href="#">vtss_phy_statistic_t</a>	Phy statistic information . . . . .	357
<a href="#">vtss_phy_status_1g_t</a>	PHY 1G status . . . . .	359
<a href="#">vtss_phy_tbi_conf_t</a>	PHY TBI configuration . . . . .	360
<a href="#">vtss_phy_timestamp_t</a>	PHY timestamp in seconds and nanoseconds (10 bytes Timestamp) . . . . .	361
<a href="#">vtss_phy_ts_ach_conf_t</a>	Analyzer ACH comparator configuration options . . . . .	362
<a href="#">vtss_phy_ts_alt_clock_mode_s</a>	Parameter for setting the alternative clock mode . . . . .	364
<a href="#">vtss_phy_ts_eng_init_conf_t</a>	Defines the basic engine parameters passed to the engine_init_conf_get() function . . . . .	365
<a href="#">vtss_phy_ts_engine_action_t</a>	Engine Action configuration options . . . . .	367
<a href="#">vtss_phy_ts_engine_flow_conf_t</a>	Analyzer flow configuration options . . . . .	369
<a href="#">vtss_phy_ts_eth_conf_t</a>	Analyzer Ethernet comparator configuration options . . . . .	371
<a href="#">vtss_phy_ts_fifo_conf_t</a>	Defines the params for FIFO SYNC function . . . . .	377
<a href="#">vtss_phy_ts_fifo_sig_t</a>	Tx TSFIFO entry signature . . . . .	378
<a href="#">vtss_phy_ts_gen_conf_t</a>	Analyzer Generic data configuration options using IP comparator . . . . .	380

<a href="#">vtss_phy_ts_generic_action_t</a>	Generic Action configuration option . . . . .	382
<a href="#">vtss_phy_ts_generic_flow_conf_t</a>	Generic engine flow configuration options . . . . .	384
<a href="#">vtss_phy_ts_ietf_mpls_ach_oam_conf_t</a>	Analyzer OAM comparator, IETF MPLS ACH OAM Packet format configuration options . . . . .	385
<a href="#">vtss_phy_ts_init_conf_t</a>	Defines the initial parameters to be passed to init function . . . . .	386
<a href="#">vtss_phy_ts_ip_conf_t</a>	Analyzer IP comparator configuration options . . . . .	390
<a href="#">vtss_phy_ts_mpls_conf_t</a>	Analyzer MPLS comparator configuration options . . . . .	394
<a href="#">vtss_phy_ts_mpls_lvl_rng_t</a>	MPLS level range . . . . .	398
<a href="#">vtss_phy_ts_nphase_status_t</a>	N-phase status . . . . .	399
<a href="#">vtss_phy_ts_oam_engine_action_t</a>	OAM Action configuration options . . . . .	400
<a href="#">vtss_phy_ts_oam_engine_flow_conf_t</a>	OAM engine flow configuration options . . . . .	402
<a href="#">vtss_phy_ts_pps_config_s</a>	PPS Configuration . . . . .	403
<a href="#">vtss_phy_ts_ptp_conf_t</a>	Analyzer PTP comparator configuration options . . . . .	405
<a href="#">vtss_phy_ts_ptp_engine_action_t</a>	Analyzer PTP action configuration options . . . . .	407
<a href="#">vtss_phy_ts_ptp_engine_flow_conf_t</a>	PTP engine flow configuration options . . . . .	408
<a href="#">vtss_phy_ts_sertod_conf_t</a>	PHY timestamp in seconds and nanoseconds (10 bytes Timestamp) . . . . .	410
<a href="#">vtss_phy_ts_stats_t</a>	Timestamping Statistics . . . . .	411
<a href="#">vtss_phy_ts_y1731_oam_conf_t</a>	Analyzer OAM comparator, Y.1731 OAM Packet format configuration options . . . . .	414
<a href="#">vtss_phy_type_t</a>	Phy type information . . . . .	416
<a href="#">vtss_phy_veriphy_result_t</a>	VeriPHY result . . . . .	418
<a href="#">vtss_phy_wol_conf_t</a>	Structure for Get/Set Wake-On-LAN configuration . . . . .	419
<a href="#">vtss_pi_conf_t</a>	PI configuration . . . . .	420
<a href="#">vtss_policer_ext_t</a>	Policer Extensions . . . . .	421
<a href="#">vtss_policer_t</a>	Policer . . . . .	425
<a href="#">vtss_port_bridge_counters_t</a>	Port bridge counter structure (RFC 4188) . . . . .	425
<a href="#">vtss_port_clause_37_adv_t</a>	Advertisement control data for Clause 37 aneg . . . . .	426
<a href="#">vtss_port_clause_37_control_t</a>	Auto-negotiation control parameter struct . . . . .	428
<a href="#">vtss_port_conf_t</a>	Port configuration structure . . . . .	429
<a href="#">vtss_port_counters_t</a>	Port counter structure . . . . .	433
<a href="#">vtss_port_ethernet_like_counters_t</a>	Ethernet-like Interface counter structure (RFC 3635) . . . . .	435

<a href="#">vtss_port_flow_control_conf_t</a>	Flow control setup . . . . .	438
<a href="#">vtss_port_frame_gaps_t</a>	Inter frame gap structure . . . . .	440
<a href="#">vtss_port_if_group_counters_t</a>	Interfaces Group counter structure (RFC 2863) . . . . .	441
<a href="#">vtss_port_ifh_t</a>	Port Internal Frame Header structure . . . . .	444
<a href="#">vtss_port_map_t</a>	Port map structure . . . . .	445
<a href="#">vtss_port_proprietary_counters_t</a>	Port proprietary counter structure . . . . .	447
<a href="#">vtss_port_rmon_counters_t</a>	RMON counter structure (RFC 2819) . . . . .	448
<a href="#">vtss_port_serdes_conf_t</a>	SFI Serdes configuration . . . . .	455
<a href="#">vtss_port_sgmii_aneg_t</a>	Advertisement control data for SGMII aneg . . . . .	456
<a href="#">vtss_port_status_t</a>	Port status parameter struct . . . . .	458
<a href="#">vtss_qce_action_t</a>	QCE action . . . . .	460
<a href="#">vtss_qce_frame_etype_t</a>	Frame data for VTSS_QCE_TYPE_ETYPE . . . . .	462
<a href="#">vtss_qce_frame_ipv4_t</a>	Frame data for VTSS_QCE_TYPE_IPV4 . . . . .	463
<a href="#">vtss_qce_frame_ipv6_t</a>	Frame data for VTSS_QCE_TYPE_IPV6 . . . . .	465
<a href="#">vtss_qce_frame_llc_t</a>	Frame data for VTSS_QCE_TYPE_LLC . . . . .	466
<a href="#">vtss_qce_frame_snap_t</a>	Frame data for VTSS_QCE_TYPE_SNAP . . . . .	467
<a href="#">vtss_qce_key_t</a>	QCE key . . . . .	468
<a href="#">vtss_qce_mac_t</a>	QCE MAC information . . . . .	470
<a href="#">vtss_qce_t</a>	QoS Control Entry . . . . .	472
<a href="#">vtss_qce_tag_t</a>	QCE tag information . . . . .	473
<a href="#">vtss_qos_conf_t</a>	All parameters below are defined per chip . . . . .	474
<a href="#">vtss_qos_port_conf_t</a>	QoS setup per port . . . . .	476
<a href="#">vtss_qs_conf_t</a>	Queue System settings . . . . .	482
<a href="#">vtss_rcpll_status_t</a>	Structure for Get PHY RC-PLL status . . . . .	484
<a href="#">vtss_red_t</a>	Random Early Detection configuration struct version 1 (per port, per queue) . . . . .	486
<a href="#">vtss_restart_status_t</a>	Restart status . . . . .	487
<a href="#">vtss_routing_entry_t</a>	Routing entry . . . . .	489
<a href="#">vtss_secure_on_passwd_t</a>	Structure for Wake-On-LAN Secure-On Password . . . . .	490
<a href="#">vtss_serdes_macro_conf_t</a>	Serdes macro configuration . . . . .	491

<code>vtss_sflow_port_conf_t</code>	SFlow configuration structure . . . . .	492
<code>vtss_sgpi_o_conf_t</code>	SGPIO configuration for a group . . . . .	493
<code>vtss_sgpi_o_port_conf_t</code>	SGPIO port configuration . . . . .	494
<code>vtss_sgpi_o_port_data_t</code>	SGPIO read data for a port . . . . .	496
<code>vtss_shaper_t</code>	Shaper . . . . .	496
<code>vtss_sublayer_status_t</code>	10G Phy link and fault status . . . . .	497
<code>vtss_sync_e_clock_in_t</code>	Struct containing configuration selecting the recovered input clock port, to be delivered to a selected output clock port . . . . .	499
<code>vtss_sync_e_clock_out_t</code>	Struct containing configuration for a recovered clock output port . . . . .	500
<code>vtss_tci_t</code>	Tag Control Information (according to IEEE 802.1Q) . . . . .	501
<code>vtss_timeofday_t</code>	Time of day structure . . . . .	502
<code>vtss_timestamp_t</code>	Time stamp in seconds and nanoseconds . . . . .	503
<code>vtss_trace_conf_t</code>	Trace group configuration . . . . .	504
<code>vtss_ts_ext_clock_mode_t</code>	External clock output configuration . . . . .	505
<code>vtss_ts_id_t</code>	Timestamp identifier . . . . .	506
<code>vtss_ts_internal_mode_t</code>	Hardware timestamping format mode for internal ports . . . . .	507
<code>vtss_ts_operation_mode_t</code>	Timestamp operation . . . . .	508
<code>vtss_ts_timestamp_alloc_t</code>	Timestamp allocation . . . . .	508
<code>vtss_ts_timestamp_t</code>	Timestamp structure . . . . .	509
<code>vtss_vcap_ip_t</code>	VCAP IPv4 address value and mask . . . . .	511
<code>vtss_vcap_u128_t</code>	VCAP 128 bit value and mask . . . . .	512
<code>vtss_vcap_u16_t</code>	VCAP 16 bit value and mask . . . . .	513
<code>vtss_vcap_u24_t</code>	VCAP 24 bit value and mask . . . . .	514
<code>vtss_vcap_u32_t</code>	VCAP 32 bit value and mask . . . . .	515
<code>vtss_vcap_u40_t</code>	VCAP 40 bit value and mask . . . . .	516
<code>vtss_vcap_u48_t</code>	VCAP 48 bit value and mask . . . . .	517
<code>vtss_vcap_u8_t</code>	VCAP 8 bit value and mask . . . . .	518
<code>vtss_vcap_udp_tcp_t</code>	VCAP UDP/TCP port range . . . . .	519
<code>vtss_vcap_vid_t</code>	VCAP VLAN ID value and mask . . . . .	520

<a href="#">vtss_vcap_vr_t</a>	VCAP universal value or range . . . . .	521
<a href="#">vtss_vce_action_t</a>	VCE Action . . . . .	523
<a href="#">vtss_vce_frame_etype_t</a>	Frame data for VTSS_VCE_TYPE_ETYPE . . . . .	524
<a href="#">vtss_vce_frame_ipv4_t</a>	Frame data for VTSS_VCE_TYPE_IPV4 . . . . .	525
<a href="#">vtss_vce_frame_ipv6_t</a>	Frame data for VTSS_VCE_TYPE_IPV6 . . . . .	527
<a href="#">vtss_vce_frame_llc_t</a>	Frame data for VTSS_VCE_TYPE_LLCC . . . . .	528
<a href="#">vtss_vce_frame_snap_t</a>	Frame data for VTSS_VCE_TYPE_SNAP . . . . .	529
<a href="#">vtss_vce_key_t</a>	VCE Key . . . . .	530
<a href="#">vtss_vce_mac_t</a>	VCE MAC header information . . . . .	532
<a href="#">vtss_vce_t</a>	VLAN Control Entry . . . . .	533
<a href="#">vtss_vce_tag_t</a>	VCE tag information . . . . .	534
<a href="#">vtss_vcl_port_conf_t</a>	VCL port configuration . . . . .	536
<a href="#">vtss_vid_mac_t</a>	MAC Address in specific VLAN . . . . .	537
<a href="#">vtss_vlan_conf_t</a>	VLAN configuration . . . . .	538
<a href="#">vtss_vlan_port_conf_t</a>	VLAN port configuration . . . . .	539
<a href="#">vtss_vlan_tag_t</a>	. . . . .	540
<a href="#">vtss_vlan_trans_grp2vlan_conf_t</a>	VLAN translation group-to-VLAN configuration . . . . .	542
<a href="#">vtss_vlan_trans_port2grp_conf_t</a>	VLAN translation port-to-group configuration . . . . .	543
<a href="#">vtss_vlan_vid_conf_t</a>	VLAN ID configuration . . . . .	544
<a href="#">vtss_vstax_conf_t</a>	VStaX configuration for switch . . . . .	545
<a href="#">vtss_vstax_glag_entry_t</a>	GLAG info . . . . .	547
<a href="#">vtss_vstax_port_conf_t</a>	VStaX setup for port . . . . .	547
<a href="#">vtss_vstax_route_entry_t</a>	UPSID Route Entry . . . . .	548
<a href="#">vtss_vstax_route_table_t</a>	UPSID Route Table . . . . .	549
<a href="#">vtss_vstax_rx_header_t</a>	VStaX frame header used for reception . . . . .	550
<a href="#">vtss_vstax_tx_header_t</a>	VStaX frame header used for transmission . . . . .	552
<a href="#">vtss_wol_mac_addr_t</a>	Structure for Wake-On-LAN MAC Address . . . . .	555



# Chapter 6

## File Index

### 6.1 File List

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

vtss_api/include/ <a href="#">vtss_ae_api.h</a>	
Ae API	630
vtss_api/include/ <a href="#">vtss_afi_api.h</a>	
AFI API	630
vtss_api/include/ <a href="#">vtss_aneg_api.h</a>	
ANEG API	631
vtss_api/include/ <a href="#">vtss_api.h</a>	
Vitesse API main header file	631
vtss_api/include/ <a href="#">vtss_evc_api.h</a>	
EVC API	631
vtss_api/include/ <a href="#">vtss_fdma_api.h</a>	
Frame DMA API	642
vtss_api/include/ <a href="#">vtss_gfp_api.h</a>	
GFP API	659
vtss_api/include/ <a href="#">vtss_hqos_api.h</a>	
HQoS API	659
vtss_api/include/ <a href="#">vtss_i2c_api.h</a>	
I2C API	659
vtss_api/include/ <a href="#">vtss_init_api.h</a>	
Initialization API	660
vtss_api/include/ <a href="#">vtss_l2_api.h</a>	
Layer 2 API	676
vtss_api/include/ <a href="#">vtss_l3_api.h</a>	
L3 routing API	741
vtss_api/include/ <a href="#">vtss_mac10g_api.h</a>	
MAC10G API	753
vtss_api/include/ <a href="#">vtss_macsec_api.h</a>	
MACsec API	??
vtss_api/include/ <a href="#">vtss_misc_api.h</a>	
Miscellaneous API	753
vtss_api/include/ <a href="#">vtss_mpls_api.h</a>	
MPLS API	785
vtss_api/include/ <a href="#">vtss_oam_api.h</a>	
OAM API	785
vtss_api/include/ <a href="#">vtss_oha_api.h</a>	
OHA API	786

vtss_api/include/vtss_os.h	786
OS Layer API . . . . .	786
vtss_api/include/vtss_os_custom.h	786
OS custom header file . . . . .	786
vtss_api/include/vtss_os_ecos.h	791
ECos OS API . . . . .	791
vtss_api/include/vtss_os_linux.h	801
Linux OS API . . . . .	801
vtss_api/include/vtss_otn_api.h	808
OTN API . . . . .	808
vtss_api/include/vtss_packet_api.h	808
Packet API . . . . .	808
vtss_api/include/vtss_pcs_10gbase_r_api.h	831
PCS_10BASE_R API . . . . .	831
vtss_api/include/vtss_phy_10g_api.h	831
10G PHY API . . . . .	831
vtss_api/include/vtss_phy_api.h	926
PHY API . . . . .	926
vtss_api/include/vtss_phy_ts_api.h	1003
PHY TimeStamping API . . . . .	1003
vtss_api/include/vtss_port_api.h	1075
Port API . . . . .	1075
vtss_api/include/vtss_qos_api.h	1091
QoS API . . . . .	1091
vtss_api/include/vtss_rab_api.h	1099
RAB API . . . . .	1099
vtss_api/include/vtss_security_api.h	1100
Security API . . . . .	1100
vtss_api/include/vtss_sfi4_api.h	1109
SFI4 API . . . . .	1109
vtss_api/include/vtss_sync_api.h	1109
Synchronization API . . . . .	1109
vtss_api/include/vtss_tfi5_api.h	1113
TFI5 API . . . . .	1113
vtss_api/include/vtss_ts_api.h	1113
TimeStamping API . . . . .	1113
vtss_api/include/vtss_upi_api.h	1131
Define UPI API interface . . . . .	1131
vtss_api/include/vtss_wis_api.h	1131
EWIS layer API . . . . .	1131
vtss_api/include/vtss_xaui_api.h	1163
XAUI API . . . . .	1163
vtss_api/include/vtss_xfi_api.h	1163
XFI API . . . . .	1163
vtss_api/include/vtss/api/l2_types.h	557
Layer 2 Public API Header for l2 . . . . .	557
vtss_api/include/vtss/api/options.h	558
Features and options . . . . .	558
vtss_api/include/vtss/api/phy.h	578
PHY Public API Header . . . . .	578
vtss_api/include/vtss/api/port.h	580
Port Public API Header . . . . .	580
vtss_api/include/vtss/api/types.h	592
Generic types API . . . . .	592

## 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`
- `vtss_phy_power_mode_t power_mode`
- `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 power\_mode

```
vtss_phy_power_mode_t port_custom_conf_t::power_mode
```

PHY power mode

Definition at line 281 of file port.h.

### 7.2.2.10 exc\_col\_cont

```
BOOL port_custom_conf_t::exc_col_cont
```

Excessive collision continuation

Definition at line 283 of file port.h.

#### 7.2.2.11 adv\_dis

`u8 port_custom_conf_t::adv_dis`

Auto neg advertisement disable

Definition at line 284 of file port.h.

#### 7.2.2.12 max\_tags

`u8 port_custom_conf_t::max_tags`

Maximum number of tags

Definition at line 285 of file port.h.

#### 7.2.2.13 oper\_up

`BOOL port_custom_conf_t::oper_up`

Force operational state up

Definition at line 286 of file port.h.

#### 7.2.2.14 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 `tag_vtss_fdma_list` Struct Reference

Software DCB structure.

```
#include <vtss_fdma_api.h>
```

## Data Fields

- `u8 * frm_ptr`
- `u32 act_len`
- `void * alloc_ptr`
- `vtss_packet_rx_info_t * rx_info`
- `VTSS_OS_TIMESTAMP_TYPE sw_tstamp`
- `void * user`
- `u64 afi_frm_cnt`
- `u32 afi_seq_number`
- `struct tag_vtss_fdma_list * next`

### 7.4.1 Detailed Description

Software DCB structure.

The following structure defines a software DCB. Software DCBs can be linked together to form a list of software DCBs. This is the fundamental data structure used to transfer information between a user-level application and the FDMA driver code.

The structure holds the actual hardware DCB needed by the FDMA silicon, a pointer to the associated data area, and other properties such as frame length and allocation length.

The structure is used for both injection and extraction, but some of its members' meaning change slightly in the two cases. The exact interpretation is shown as comments inside the definition below.

Definition at line 347 of file vtss\_fdma\_api.h.

### 7.4.2 Field Documentation

#### 7.4.2.1 frm\_ptr

`u8* tag_vtss_fdma_list::frm_ptr`

##### XTR:

This points to the first byte of the frame. Set by FDMA driver.

For SOF DCBs, this corresponds to the first byte of the DMAC. For non-SOF DCBs it points to the first byte of the continued frame.

##### INJ/AFI:

This points to the first byte of the frame. Set by application. For SOF DCBs, VTSS\_FDMA\_HDR\_SIZE\_BYTES of head room must be available just before the `frm_ptr`. The FDMA driver may update this internally, so don't rely on its value after injection.

Definition at line 416 of file vtss\_fdma\_api.h.

#### 7.4.2.2 act\_len

```
u32 tag_vtss_fdma_list::act_len
```

**XTR:**

Used internally by the FDMA driver (holds length incl. IFH, frame, and FCS). **INJ/AFI:**

The number of frame bytes to be injected from [frm\\_ptr](#) for this fragment. For the SOF DCB, it does not include the size of IFH - only true frame data. for the EOF DCB, it does not include the size of the FCS. The FDMA driver may update this internally, so don't rely on its value after injection.

Definition at line 442 of file vtss\_fdma\_api.h.

#### 7.4.2.3 alloc\_ptr

```
void* tag_vtss_fdma_list::alloc_ptr
```

**XTR:**

Pointer to allocated frame + meta data. Either set by application during calls to rx\_alloc\_cb() or by the FDMA driver itself if memory management is entirely handled by the FDMA driver. **INJ/AFI:**

Not used.

Definition at line 455 of file vtss\_fdma\_api.h.

#### 7.4.2.4 rx\_info

```
vtss_packet_rx_info_t* tag_vtss_fdma_list::rx_info
```

**XTR:**

Pointer to decoded extraction header. The allocation of this is taken care of by the FDMA driver. Only valid in SOF DCB. **INJ/AFI:**

Not used.

Definition at line 466 of file vtss\_fdma\_api.h.

#### 7.4.2.5 sw\_tstamp

```
VTSS_OS_TIMESTAMP_TYPE tag_vtss_fdma_list::sw_tstamp
```

**XTR:**

Unused. In V3+, it's part of the [vtss\\_packet\\_rx\\_info\\_t](#) structure and is called sw\_tstamp. **INJ:**

The FDMA driver code time-stamps the packet when the [vtss\\_fdma\\_irq\\_handler\(\)](#) gets invoked based on an injection interrupt.

**. AFI:**

Unused. The FDMA driver is agnostic to the time stamp format, and it's up to the platform header ([vtss\\_os.h](#)) to define appropriate types and functions for obtaining the time stamp.

Definition at line 509 of file vtss\_fdma\_api.h.

## 7.4.2.6 user

```
void* tag_vtss_fdma_list::user
```

**XTR/INJ/AFI:**

A pointer to any user data. Set by user and used only by the user. The FDMA code doesn't touch nor uses it.

Definition at line 516 of file vtss\_fdma\_api.h.

## 7.4.2.7 afi\_frm\_cnt

```
u64 tag_vtss_fdma_list::afi_frm_cnt
```

**XTR/INJ**

Not used. **AFI**

Output parameter. Holds the number of frames that was actually transmitted. Updated regularly, but is only 100% correct once the AFI injection is cancelled and the tx\_done\_cb() is called.

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

Definition at line 684 of file vtss\_fdma\_api.h.

## 7.4.2.8 afi\_seq\_number

```
u32 tag_vtss_fdma_list::afi_seq_number
```

**XTR/INJ**

Not used. **AFI**

Holds the next sequence number to put in a given frame. Updated repeatedly by S/W when AFI sequence numbering is enabled.

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

Definition at line 701 of file vtss\_fdma\_api.h.

## 7.4.2.9 next

```
struct tag_vtss_fdma_list* tag_vtss_fdma_list::next
```

**XTR:**

Points to the next entry in the list or NULL if it's the last. Set by user on initialization of list. Continuously updated by vtss\_fdma.c afterwards.

**INJ:**

Points to the next fragment of the frame and set by user on a per-frame basis. Last fragment of a frame must set ->next to NULL. Once handed to vtss\_fdma.c, the driver code takes over. **AFI:**

Must be NULL (AFI frames must be contained in one fragment (due to injection from multiple GPDMA channels into the same injection group)). Internally the FDMA driver uses it to link multiple user AFI frames onto the same AFI channel.

Definition at line 717 of file vtss\_fdma\_api.h.

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

- vtss\_api/include/vtss\_fdma\_api.h

## 7.5 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](#)

### 7.5.1 Detailed Description

Frame data for VTSS\_ACE\_TYPE\_ARP.

Definition at line 450 of file vtss\_security\_api.h.

### 7.5.2 Field Documentation

#### 7.5.2.1 smac

```
vtss_ace_u48_t vtss_ace_frame_arp_t::smac
```

SMAC

Definition at line 452 of file vtss\_security\_api.h.

#### 7.5.2.2 arp

```
vtss_ace_bit_t vtss_ace_frame_arp_t::arp
```

Opcode ARP/RARP

Definition at line 453 of file vtss\_security\_api.h.

### 7.5.2.3 req

`vtss_ace_bit_t` `vtss_ace_frame_arp_t::req`

Opcode request/reply

Definition at line 454 of file vtss\_security\_api.h.

### 7.5.2.4 unknown

`vtss_ace_bit_t` `vtss_ace_frame_arp_t::unknown`

Opcode unknown

Definition at line 455 of file vtss\_security\_api.h.

### 7.5.2.5 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.

### 7.5.2.6 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.

### 7.5.2.7 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.

### 7.5.2.8 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`.

### 7.5.2.9 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`.

### 7.5.2.10 sip

`vtss_ace_ip_t vtss_ace_frame_arp_t::sip`

Sender IP address

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

### 7.5.2.11 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`

## 7.6 `vtss_ace_frame_etype_t` Struct Reference

Frame data for `VTSS_ACE_TYPEETYPE`.

```
#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`

### 7.6.1 Detailed Description

Frame data for VTSS\_ACE\_TYPE\_ETYPE.

Definition at line 422 of file vtss\_security\_api.h.

### 7.6.2 Field Documentation

#### 7.6.2.1 dmac

`vtss_ace_u48_t vtss_ace_frame_etype_t::dmac`

DMAC

Definition at line 424 of file vtss\_security\_api.h.

#### 7.6.2.2 smac

`vtss_ace_u48_t vtss_ace_frame_etype_t::smac`

SMAC

Definition at line 425 of file vtss\_security\_api.h.

#### 7.6.2.3 etype

`vtss_ace_u16_t vtss_ace_frame_etype_t::etype`

Ethernet Type value

Definition at line 426 of file vtss\_security\_api.h.

#### 7.6.2.4 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`

## 7.7 `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`

### 7.7.1 Detailed Description

Frame data for VTSS\_ACE\_TYPE\_IPV4.

Definition at line 466 of file `vtss_security_api.h`.

### 7.7.2 Field Documentation

### 7.7.2.1 ttl

`vtss_ace_bit_t` `vtss_ace_frame_ipv4_t::ttl`

TTL zero

Definition at line 468 of file vtss\_security\_api.h.

### 7.7.2.2 fragment

`vtss_ace_bit_t` `vtss_ace_frame_ipv4_t::fragment`

Fragment

Definition at line 469 of file vtss\_security\_api.h.

### 7.7.2.3 options

`vtss_ace_bit_t` `vtss_ace_frame_ipv4_t::options`

Header options

Definition at line 470 of file vtss\_security\_api.h.

### 7.7.2.4 ds

`vtss_ace_u8_t` `vtss_ace_frame_ipv4_t::ds`

DS field

Definition at line 471 of file vtss\_security\_api.h.

### 7.7.2.5 proto

`vtss_ace_u8_t` `vtss_ace_frame_ipv4_t::proto`

Protocol

Definition at line 472 of file vtss\_security\_api.h.

### 7.7.2.6 sip

`vtss_ace_ip_t` `vtss_ace_frame_ipv4_t::sip`

Source IP address

Definition at line 473 of file vtss\_security\_api.h.

### 7.7.2.7 dip

`vtss_ace_ip_t` `vtss_ace_frame_ipv4_t::dip`

Destination IP address

Definition at line 474 of file vtss\_security\_api.h.

### 7.7.2.8 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.

### 7.7.2.9 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.

### 7.7.2.10 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.

### 7.7.2.11 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.

### 7.7.2.12 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.

### 7.7.2.13 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.

### 7.7.2.14 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.

### 7.7.2.15 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.

### 7.7.2.16 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.

### 7.7.2.17 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.

### 7.7.2.18 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.

### 7.7.2.19 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](#)

## 7.8 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`

### 7.8.1 Detailed Description

Frame data for VTSS\_ACE\_TYPE\_IPV6.

Definition at line 494 of file vtss\_security\_api.h.

### 7.8.2 Field Documentation

#### 7.8.2.1 proto

`vtss_ace_u8_t vtss_ace_frame_ipv6_t::proto`

IPv6 protocol

Definition at line 496 of file vtss\_security\_api.h.

#### 7.8.2.2 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.

### 7.8.2.3 ttl

`vtss_ace_bit_t vtss_ace_frame_ipv6_t::ttl`

TTL zero

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

### 7.8.2.4 ds

`vtss_ace_u8_t vtss_ace_frame_ipv6_t::ds`

DS field

Definition at line 499 of file `vtss_security_api.h`.

### 7.8.2.5 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`.

### 7.8.2.6 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`.

### 7.8.2.7 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`.

### 7.8.2.8 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.

### 7.8.2.9 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.

### 7.8.2.10 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.

### 7.8.2.11 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.

### 7.8.2.12 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.

#### 7.8.2.13 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`.

#### 7.8.2.14 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`.

#### 7.8.2.15 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`.

#### 7.8.2.16 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`

## 7.9 `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`

### 7.9.1 Detailed Description

Frame data for VTSS\_ACE\_TYPE LLC.

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

### 7.9.2 Field Documentation

#### 7.9.2.1 dmac

`vtss_ace_u48_t vtss_ace_frame_llc_t::dmac`

DMAC

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

#### 7.9.2.2 smac

`vtss_ace_u48_t vtss_ace_frame_llc_t::smac`

SMAC

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

#### 7.9.2.3 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`

## 7.10 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`

#### 7.10.1 Detailed Description

Frame data for VTSS\_ACE\_TYPE\_SNAP.

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

#### 7.10.2 Field Documentation

##### 7.10.2.1 dmac

```
vtss_ace_u48_t vtss_ace_frame_snap_t::dmac
```

DMAC

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

##### 7.10.2.2 smac

```
vtss_ace_u48_t vtss_ace_frame_snap_t::smac
```

SMAC

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

### 7.10.2.3 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](#)

## 7.11 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](#)

### 7.11.1 Detailed Description

Access Control Entry.

Definition at line 518 of file vtss\_security\_api.h.

### 7.11.2 Field Documentation

#### 7.11.2.1 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.

#### 7.11.2.2 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.

#### 7.11.2.3 policy

`vtss_ace_u8_t` `vtss_ace_t::policy`

Policy number

Definition at line 532 of file vtss\_security\_api.h.

#### 7.11.2.4 type

`vtss_ace_type_t` `vtss_ace_t::type`

ACE frame type

Definition at line 533 of file vtss\_security\_api.h.

#### 7.11.2.5 action

`vtss_acl_action_t` `vtss_ace_t::action`

ACE action

Definition at line 534 of file vtss\_security\_api.h.

### 7.11.2.6 dmac\_mc

`vtss_ace_bit_t` `vtss_ace_t::dmac_mc`

Multicast DMAC

Definition at line 536 of file vtss\_security\_api.h.

### 7.11.2.7 dmac\_bc

`vtss_ace_bit_t` `vtss_ace_t::dmac_bc`

Broadcast DMAC

Definition at line 537 of file vtss\_security\_api.h.

### 7.11.2.8 vlan

`vtss_ace_vlan_t` `vtss_ace_t::vlan`

VLAN Tag

Definition at line 539 of file vtss\_security\_api.h.

### 7.11.2.9 etype

`vtss_ace_frame_etype_t` `vtss_ace_t::etype`

VTSS\_ACE\_TYPE\_ETYPE

Definition at line 544 of file vtss\_security\_api.h.

### 7.11.2.10 llc

`vtss_ace_frame_llc_t` `vtss_ace_t::llc`

VTSS\_ACE\_TYPE\_LL

Definition at line 545 of file vtss\_security\_api.h.

### 7.11.2.11 snap

`vtss_ace_frame_snap_t` `vtss_ace_t::snap`

VTSS\_ACE\_TYPE\_SNAP

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

### 7.11.2.12 arp

`vtss_ace_frame_arp_t` `vtss_ace_t::arp`

VTSS\_ACE\_TYPE\_ARP

Definition at line 547 of file `vtss_security_api.h`.

### 7.11.2.13 ipv4

`vtss_ace_frame_ipv4_t` `vtss_ace_t::ipv4`

VTSS\_ACE\_TYPE\_IPV4

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

### 7.11.2.14 ipv6

`vtss_ace_frame_ipv6_t` `vtss_ace_t::ipv6`

VTSS\_ACE\_TYPE\_IPV6

Definition at line 549 of file `vtss_security_api.h`.

### 7.11.2.15 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`

## 7.12 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`

#### 7.12.1 Detailed Description

ACE VLAN information.

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

#### 7.12.2 Field Documentation

##### 7.12.2.1 vid

`vtss_ace_vid_t vtss_ace_vlan_t::vid`

VLAN ID (12 bit)

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

##### 7.12.2.2 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`.

### 7.12.2.3 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`

## 7.13 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`

### 7.13.1 Detailed Description

ACL Action.

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

### 7.13.2 Field Documentation

#### 7.13.2.1 cpu

`BOOL vtss_acl_action_t::cpu`

Forward to CPU

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

### 7.13.2.2 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.

### 7.13.2.3 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.

### 7.13.2.4 police

`BOOL vtss_acl_action_t::police`

Enable policer

Definition at line 243 of file vtss\_security\_api.h.

### 7.13.2.5 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.

### 7.13.2.6 learn

`BOOL vtss_acl_action_t::learn`

Allow learning

Definition at line 251 of file vtss\_security\_api.h.

### 7.13.2.7 forward

`BOOL vtss_acl_action_t::forward`

Allow forwarding

Definition at line 253 of file `vtss_security_api.h`.

### 7.13.2.8 port\_forward

`BOOL vtss_acl_action_t::port_forward`

Forward to specific port

Definition at line 254 of file `vtss_security_api.h`.

### 7.13.2.9 port\_no

`vtss_port_no_t vtss_acl_action_t::port_no`

Specified port

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

### 7.13.2.10 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`

## 7.14 vtss\_acl\_policer\_conf\_t Struct Reference

ACL policer configuration.

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

## Data Fields

- [vtss\\_packet\\_rate\\_t rate](#)

### 7.14.1 Detailed Description

ACL policer configuration.

Definition at line 155 of file vtss\_security\_api.h.

### 7.14.2 Field Documentation

#### 7.14.2.1 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](#)

## 7.15 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](#)

### 7.15.1 Detailed Description

ACL port configuration.

Definition at line 276 of file vtss\_security\_api.h.

### 7.15.2 Field Documentation

#### 7.15.2.1 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`.

#### 7.15.2.2 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`

## 7.16 `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`

#### 7.16.1 Detailed Description

Aggregation traffic distribution mode.

Definition at line 39 of file `l2_types.h`.

## 7.16.2 Field Documentation

### 7.16.2.1 smac\_enable

`BOOL vtss_aggr_mode_t::smac_enable`

Source MAC address

Definition at line 41 of file l2\_types.h.

### 7.16.2.2 dmac\_enable

`BOOL vtss_aggr_mode_t::dmac_enable`

Destination MAC address

Definition at line 42 of file l2\_types.h.

### 7.16.2.3 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.

### 7.16.2.4 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](#)

## 7.17 vtss\_aneg\_t Struct Reference

Auto negotiation struct.

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

### Data Fields

- `BOOL obey_pause`
- `BOOL generate_pause`

#### 7.17.1 Detailed Description

Auto negotiation struct.

Definition at line 482 of file types.h.

#### 7.17.2 Field Documentation

##### 7.17.2.1 obey\_pause

```
BOOL vtss_aneg_t::obey_pause
```

This port should obey PAUSE frames

Definition at line 484 of file types.h.

##### 7.17.2.2 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`

## 7.18 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`

### 7.18.1 Detailed Description

API lock structure.

Definition at line 285 of file vtss\_misc\_api.h.

### 7.18.2 Field Documentation

#### 7.18.2.1 inst

`vtss_inst_t vtss_api_lock_t::inst`

Target instance reference

Definition at line 286 of file vtss\_misc\_api.h.

#### 7.18.2.2 function

`const char* vtss_api_lock_t::function`

Function name

Definition at line 287 of file vtss\_misc\_api.h.

#### 7.18.2.3 file

`const char* vtss_api_lock_t::file`

File name

Definition at line 288 of file vtss\_misc\_api.h.

#### 7.18.2.4 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](#)

## 7.19 vtss\_basic\_counters\_t Struct Reference

Basic counters structure.

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

### Data Fields

- [u32 rx\\_frames](#)
- [u32 tx\\_frames](#)

#### 7.19.1 Detailed Description

Basic counters structure.

Definition at line 377 of file [vtss\\_port\\_api.h](#).

#### 7.19.2 Field Documentation

##### 7.19.2.1 rx\_frames

[u32](#) vtss\_basic\_counters\_t::rx\_frames

Rx frames

Definition at line 379 of file [vtss\\_port\\_api.h](#).

### 7.19.2.2 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`

## 7.20 vtss\_chip\_id\_t Struct Reference

Chip ID.

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

### Data Fields

- `u16 part_number`
- `u16 revision`

### 7.20.1 Detailed Description

Chip ID.

Definition at line 401 of file `vtss_misc_api.h`.

### 7.20.2 Field Documentation

#### 7.20.2.1 part\_number

`u16 vtss_chip_id_t::part_number`

BCD encoded part number

Definition at line 403 of file `vtss_misc_api.h`.

### 7.20.2.2 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](#)

## 7.21 vtss\_counter\_pair\_t Struct Reference

Counter pair.

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

### Data Fields

- [vtss\\_counter\\_t frames](#)
- [vtss\\_counter\\_t bytes](#)

### 7.21.1 Detailed Description

Counter pair.

Definition at line 1111 of file types.h.

### 7.21.2 Field Documentation

#### 7.21.2.1 frames

```
vtss\_counter\_t vtss_counter_pair_t::frames
```

Number of frames

Definition at line 1112 of file types.h.

### 7.21.2.2 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](#)

## 7.22 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](#)

### 7.22.1 Detailed Description

Debug information structure.

Definition at line 243 of file vtss\_misc\_api.h.

### 7.22.2 Field Documentation

#### 7.22.2.1 layer

`vtss_debug_layer_t vtss_debug_info_t::layer`

Layer

Definition at line 244 of file vtss\_misc\_api.h.

### 7.22.2.2 group

```
vtss_debug_group_t vtss_debug_info_t::group
```

Function group

Definition at line 245 of file vtss\_misc\_api.h.

### 7.22.2.3 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.

### 7.22.2.4 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.

### 7.22.2.5 full

```
BOOL vtss_debug_info_t::full
```

Full information dump

Definition at line 248 of file vtss\_misc\_api.h.

### 7.22.2.6 clear

```
BOOL vtss_debug_info_t::clear
```

Clear counters

Definition at line 249 of file vtss\_misc\_api.h.

### 7.22.2.7 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`

## 7.23 vtss\_debug\_lock\_t Struct Reference

API debug lock structure.

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

### Data Fields

- `vtss_chip_no_t chip_no`

### 7.23.1 Detailed Description

API debug lock structure.

Definition at line 307 of file vtss\_misc\_api.h.

### 7.23.2 Field Documentation

#### 7.23.2.1 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`

## 7.24 vtss\_dgroup\_port\_conf\_t Struct Reference

Destination group port configuration.

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

### Data Fields

- [vtss\\_dgroup\\_no\\_t dgroup\\_no](#)

#### 7.24.1 Detailed Description

Destination group port configuration.

Definition at line 1394 of file vtss\_l2\_api.h.

#### 7.24.2 Field Documentation

##### 7.24.2.1 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](#)

## 7.25 vtss\_dbb\_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](#)

### 7.25.1 Detailed Description

Dual leaky buckets policer configuration.

Definition at line 237 of file vtss\_qos\_api.h.

### 7.25.2 Field Documentation

#### 7.25.2.1 type

`vtss_policer_type_t vtss_dlb_policer_conf_t::type`

Policer type

Definition at line 238 of file vtss\_qos\_api.h.

#### 7.25.2.2 enable

`BOOL vtss_dlb_policer_conf_t::enable`

Enable/disable policer

Definition at line 239 of file vtss\_qos\_api.h.

#### 7.25.2.3 cm

`BOOL vtss_dlb_policer_conf_t::cm`

Colour Mode (TRUE means colour aware)

Definition at line 241 of file vtss\_qos\_api.h.

#### 7.25.2.4 cf

`BOOL vtss_dlb_policer_conf_t::cf`

Coupling Flag

Definition at line 243 of file vtss\_qos\_api.h.

### 7.25.2.5 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`.

### 7.25.2.6 cir

`vtss_bitrate_t vtss_dlb_policer_conf_t::cir`

Committed Information Rate

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

### 7.25.2.7 cbs

`vtss_burst_level_t vtss_dlb_policer_conf_t::cbs`

Committed Burst Size

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

### 7.25.2.8 eir

`vtss_bitrate_t vtss_dlb_policer_conf_t::eir`

Excess Information Rate

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

### 7.25.2.9 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`

## 7.26 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`

#### 7.26.1 Detailed Description

ECE action.

Definition at line 557 of file `vtss_evc_api.h`.

#### 7.26.2 Field Documentation

##### 7.26.2.1 dir

```
vtss_ece_dir_t vtss_ece_action_t::dir
```

Traffic direction

Definition at line 558 of file `vtss_evc_api.h`.

##### 7.26.2.2 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`.

### 7.26.2.3 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`.

### 7.26.2.4 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`.

### 7.26.2.5 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`.

### 7.26.2.6 evc\_id

`vtss_evc_id_t` `vtss_ece_action_t::evc_id`

EVC ID

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

### 7.26.2.7 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`

## 7.27 vtss\_ece\_frame\_ipv4\_t Struct Reference

ECE IPv4 information.

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

### Data Fields

- [vtss\\_vcap\\_vr\\_t dscp](#)

#### 7.27.1 Detailed Description

ECE IPv4 information.

Definition at line 461 of file vtss\_evc\_api.h.

#### 7.27.2 Field Documentation

##### 7.27.2.1 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](#)

## 7.28 vtss\_ece\_frame\_ipv6\_t Struct Reference

ECE IPv6 information.

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

### Data Fields

- [vtss\\_vcap\\_vr\\_t dscp](#)

### 7.28.1 Detailed Description

ECE IPv6 information.

Definition at line 476 of file vtss\_evc\_api.h.

### 7.28.2 Field Documentation

#### 7.28.2.1 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`

## 7.29 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`

### 7.29.1 Detailed Description

ECE inner tag.

Definition at line 538 of file vtss\_evc\_api.h.

### 7.29.2 Field Documentation

### 7.29.2.1 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.

### 7.29.2.2 vid

`vtss_vid_t vtss_ece_inner_tag_t::vid`

VLAN ID

Definition at line 541 of file vtss\_evc\_api.h.

### 7.29.2.3 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.

### 7.29.2.4 pcp

`vtss_tagprio_t vtss_ece_inner_tag_t::pcp`

PCP value

Definition at line 548 of file vtss\_evc\_api.h.

### 7.29.2.5 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](#)

## 7.30 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`  
} frame

### 7.30.1 Detailed Description

ECE key.

Definition at line 490 of file vtss\_evc\_api.h.

### 7.30.2 Field Documentation

#### 7.30.2.1 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.

#### 7.30.2.2 mac

```
vtss_ece_mac_t vtss_ece_key_t::mac
```

MAC header

Definition at line 493 of file vtss\_evc\_api.h.

### 7.30.2.3 tag

`vtss_ece_tag_t` `vtss_ece_key_t::tag`

Tag

Definition at line 494 of file vtss\_evc\_api.h.

### 7.30.2.4 inner\_tag

`vtss_ece_tag_t` `vtss_ece_key_t::inner_tag`

Inner tag

Definition at line 496 of file vtss\_evc\_api.h.

### 7.30.2.5 type

`vtss_ece_type_t` `vtss_ece_key_t::type`

Frame type

Definition at line 498 of file vtss\_evc\_api.h.

### 7.30.2.6 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.

### 7.30.2.7 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.

### 7.30.2.8 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](#)

## 7.31 vtss\_ece\_mac\_t Struct Reference

ECE MAC information.

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

### Data Fields

- [vtss\\_vcap\\_u48\\_t dmac](#)

#### 7.31.1 Detailed Description

ECE MAC information.

Definition at line 420 of file [vtss\\_evc\\_api.h](#).

#### 7.31.2 Field Documentation

##### 7.31.2.1 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](#)

## 7.32 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`

#### 7.32.1 Detailed Description

ECE outer tag.

Definition at line 517 of file `vtss_evc_api.h`.

#### 7.32.2 Field Documentation

##### 7.32.2.1 enable

```
BOOL vtss_ece_outer_tag_t::enable
```

Enable tag (VTSS\_ECE\_DIR\_NNI\_TO\_UNI only)

Definition at line 519 of file `vtss_evc_api.h`.

##### 7.32.2.2 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`.

### 7.32.2.3 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`.

### 7.32.2.4 pcp

`vtss_tagprio_t vtss_ece_outer_tag_t::pcp`

PCP value

Definition at line 529 of file `vtss_evc_api.h`.

### 7.32.2.5 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`

## 7.33 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`

### 7.33.1 Detailed Description

EVC Control Entry.

Definition at line 582 of file `vtss_evc_api.h`.

### 7.33.2 Field Documentation

#### 7.33.2.1 id

`vtss_ece_id_t vtss_ece_t::id`

Entry ID

Definition at line 583 of file vtss\_evc\_api.h.

#### 7.33.2.2 key

`vtss_ece_key_t vtss_ece_t::key`

ECE key

Definition at line 584 of file vtss\_evc\_api.h.

#### 7.33.2.3 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](#)

## 7.34 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](#)

### 7.34.1 Detailed Description

ECE tag information.

Definition at line 433 of file vtss\_evc\_api.h.

### 7.34.2 Field Documentation

#### 7.34.2.1 vid

`vtss_vcap_vr_t vtss_ece_tag_t::vid`

VLAN ID (12 bit)

Definition at line 435 of file vtss\_evc\_api.h.

#### 7.34.2.2 pcp

`vtss_vcap_u8_t vtss_ece_tag_t::pcp`

PCP (3 bit)

Definition at line 436 of file vtss\_evc\_api.h.

#### 7.34.2.3 dei

`vtss_vcap_bit_t vtss_ece_tag_t::dei`

DEI

Definition at line 437 of file vtss\_evc\_api.h.

#### 7.34.2.4 tagged

`vtss_vcap_bit_t vtss_ece_tag_t::tagged`

Tagged/untagged frame

Definition at line 438 of file vtss\_evc\_api.h.

### 7.34.2.5 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`

## 7.35 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`

### 7.35.1 Detailed Description

EEE port configuration.

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

### 7.35.2 Field Documentation

#### 7.35.2.1 eee\_ena

`BOOL` `vtss_eee_port_conf_t::eee_ena`

Enable EEE

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

### 7.35.2.2 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`.

### 7.35.2.3 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`.

### 7.35.2.4 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`.

### 7.35.2.5 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`

## 7.36 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`

### 7.36.1 Detailed Description

EEE port counters (JR only)

Definition at line 1246 of file `vtss_misc_api.h`.

### 7.36.2 Field Documentation

#### 7.36.2.1 `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`.

#### 7.36.2.2 `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`.

#### 7.36.2.3 `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`.

#### 7.36.2.4 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`.

#### 7.36.2.5 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`

## 7.37 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`

#### 7.37.1 Detailed Description

EEE port state (JR only)

Definition at line 1239 of file `vtss_misc_api.h`.

#### 7.37.2 Field Documentation

### 7.37.2.1 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.

### 7.37.2.2 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](#)

## 7.38 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

### 7.38.1 Detailed Description

Port protection configuration.

Definition at line 2141 of file vtss\_l2\_api.h.

### 7.38.2 Field Documentation

### 7.38.2.1 type

`vtss_eps_port_type_t vtss_eps_port_conf_t::type`

Protection type

Definition at line 2143 of file `vtss_l2_api.h`.

### 7.38.2.2 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`

## 7.39 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`

### 7.39.1 Detailed Description

EVC configuration (excluding UNIs)

Definition at line 309 of file `vtss_evc_api.h`.

### 7.39.2 Field Documentation

### 7.39.2.1 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`.

### 7.39.2.2 learning

`BOOL` `vtss_evc_conf_t::learning`

Enable/disable learning

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

### 7.39.2.3 pb

`vtss_evc_pb_conf_t` `vtss_evc_conf_t::pb`

PB specific configuration

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

### 7.39.2.4 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`

## 7.40 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`

### 7.40.1 Detailed Description

EVC/ECE counters.

Definition at line 1127 of file types.h.

### 7.40.2 Field Documentation

#### 7.40.2.1 rx\_green

`vtss_counter_pair_t vtss_evc_counters_t::rx_green`

Rx green frames/bytes

Definition at line 1128 of file types.h.

#### 7.40.2.2 rx\_yellow

`vtss_counter_pair_t vtss_evc_counters_t::rx_yellow`

Rx yellow frames/bytes

Definition at line 1129 of file types.h.

#### 7.40.2.3 rx\_red

`vtss_counter_pair_t vtss_evc_counters_t::rx_red`

Rx red frames/bytes

Definition at line 1130 of file types.h.

#### 7.40.2.4 rx\_discard

```
vtss_counter_pair_t vtss_evc_counters_t::rx_discard
```

Rx discarded frames/bytes

Definition at line 1131 of file types.h.

#### 7.40.2.5 tx\_discard

```
vtss_counter_pair_t vtss_evc_counters_t::tx_discard
```

Tx discarded frames/bytes

Definition at line 1132 of file types.h.

#### 7.40.2.6 tx\_green

```
vtss_counter_pair_t vtss_evc_counters_t::tx_green
```

Tx green frames/bytes

Definition at line 1133 of file types.h.

#### 7.40.2.7 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](#)

## 7.41 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]`

### 7.41.1 Detailed Description

PB specific EVC configuration.

Definition at line 275 of file `vtss_evc_api.h`.

### 7.41.2 Field Documentation

#### 7.41.2.1 nni

`BOOL vtss_evc_pb_conf_t::nni [VTSS_PORT_ARRAY_SIZE]`

NNI configuration

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

#### 7.41.2.2 ivid

`vtss_vid_t vtss_evc_pb_conf_t::ivid`

Internal VID

Definition at line 277 of file `vtss_evc_api.h`.

#### 7.41.2.3 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`.

#### 7.41.2.4 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.

#### 7.41.2.5 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.

#### 7.41.2.6 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](#)

## 7.42 vtss\_evc\_port\_conf\_t Struct Reference

EVC port configuration.

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

### Data Fields

- `BOOL dei_colouring`

#### 7.42.1 Detailed Description

EVC port configuration.

Definition at line 150 of file vtss\_evc\_api.h.

## 7.42.2 Field Documentation

### 7.42.2.1 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`

## 7.43 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`

### 7.43.1 Detailed Description

eWIS AIS-L consequent actions

Definition at line 77 of file vtss\_wis\_api.h.

## 7.43.2 Field Documentation

### 7.43.2.1 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.

### 7.43.2.2 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](#)

## 7.44 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`

### 7.44.1 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.

### 7.44.2 Field Documentation

#### 7.44.2.1 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.

#### 7.44.2.2 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.

#### 7.44.2.3 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.

#### 7.44.2.4 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.

#### 7.44.2.5 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.

#### 7.44.2.6 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.

#### 7.44.2.7 path\_txti

`vtss_ewis_txti_t` `vtss_ewis_conf_s::path_txti`

Path Trail Trace Identifier Configuration

Definition at line 320 of file vtss\_wis\_api.h.

#### 7.44.2.8 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.

#### 7.44.2.9 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.

#### 7.44.2.10 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.

#### 7.44.2.11 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`.

#### 7.44.2.12 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`.

#### 7.44.2.13 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`

## 7.45 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`

#### 7.45.1 Detailed Description

eWIS consequent actions

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

## 7.45.2 Field Documentation

### 7.45.2.1 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`.

### 7.45.2.2 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`.

### 7.45.2.3 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`

## 7.46 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`

### 7.46.1 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.

### 7.46.2 Field Documentation

#### 7.46.2.1 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.

#### 7.46.2.2 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.

#### 7.46.2.3 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.

#### 7.46.2.4 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.

### 7.46.2.5 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](#)

## 7.47 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`

### 7.47.1 Detailed Description

eWIS performance counter thresholds.

Definition at line 269 of file vtss\_wis\_api.h.

### 7.47.2 Field Documentation

#### 7.47.2.1 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.

#### 7.47.2.2 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`.

#### 7.47.2.3 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`.

#### 7.47.2.4 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`.

#### 7.47.2.5 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`

## 7.48 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`

### 7.48.1 Detailed Description

eWIS defects

Definition at line 153 of file vtss\_wis\_api.h.

### 7.48.2 Field Documentation

#### 7.48.2.1 dlos\_s

`BOOL vtss_ewis_defects_s::dlos_s`

Loss of signal

Definition at line 154 of file vtss\_wis\_api.h.

#### 7.48.2.2 doof\_s

`BOOL vtss_ewis_defects_s::doof_s`

Out of frame

Definition at line 155 of file vtss\_wis\_api.h.

### 7.48.2.3 dlof\_s

`BOOL vtss_ewis_defects_s::dlof_s`

Loss of frame

Definition at line 156 of file vtss\_wis\_api.h.

### 7.48.2.4 dais\_l

`BOOL vtss_ewis_defects_s::dais_l`

Line alarm indication signal

Definition at line 157 of file vtss\_wis\_api.h.

### 7.48.2.5 drdi\_l

`BOOL vtss_ewis_defects_s::drdi_l`

Line remote defect indication

Definition at line 158 of file vtss\_wis\_api.h.

### 7.48.2.6 dais\_p

`BOOL vtss_ewis_defects_s::dais_p`

Path alarm indication signal

Definition at line 159 of file vtss\_wis\_api.h.

### 7.48.2.7 dlop\_p

`BOOL vtss_ewis_defects_s::dlop_p`

Loss of pointer

Definition at line 160 of file vtss\_wis\_api.h.

#### 7.48.2.8 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.

#### 7.48.2.9 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.

#### 7.48.2.10 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.

#### 7.48.2.11 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.

#### 7.48.2.12 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.

#### 7.48.2.13 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`.

#### 7.48.2.14 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`

## 7.49 `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`

### 7.49.1 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`.

## 7.49.2 Field Documentation

### 7.49.2.1 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.

### 7.49.2.2 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.

### 7.49.2.3 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.

### 7.49.2.4 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.

### 7.49.2.5 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.

#### 7.49.2.6 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.

#### 7.49.2.7 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.

#### 7.49.2.8 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.

#### 7.49.2.9 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.

#### 7.49.2.10 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.

### 7.49.2.11 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`

## 7.50 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`

### 7.50.1 Detailed Description

eWIS force modes

Definition at line 116 of file vtss\_wis\_api.h.

### 7.50.2 Field Documentation

#### 7.50.2.1 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.

### 7.50.2.2 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](#).

### 7.50.2.3 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](#)

## 7.51 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](#)

### 7.51.1 Detailed Description

eWIS line force mode

Definition at line 98 of file [vtss\\_wis\\_api.h](#).

### 7.51.2 Field Documentation

### 7.51.2.1 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.

### 7.51.2.2 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](#)

## 7.52 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`

### 7.52.1 Detailed Description

eWIS line TX force mode

Definition at line 104 of file vtss\_wis\_api.h.

### 7.52.2 Field Documentation

### 7.52.2.1 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`.

### 7.52.2.2 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`

## 7.53 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`

### 7.53.1 Detailed Description

eWIS path force modes

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

### 7.53.2 Field Documentation

### 7.53.2.1 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.

### 7.53.2.2 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](#)

## 7.54 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](#)

### 7.54.1 Detailed Description

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

Definition at line 129 of file vtss\_wis\_api.h.

### 7.54.2 Field Documentation

#### 7.54.2.1 pn\_ebc\_mode\_s

`vtss_ewis_perf_cntr_mode_t` `vtss_ewis_perf_mode_s::pn_ebc_mode_s`

Section BIP error count (B1)

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

#### 7.54.2.2 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`.

#### 7.54.2.3 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`.

#### 7.54.2.4 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`.

#### 7.54.2.5 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`

## 7.55 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

#### 7.55.1 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.

#### 7.55.2 Field Documentation

##### 7.55.2.1 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.

##### 7.55.2.2 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.

### 7.55.2.3 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`.

### 7.55.2.4 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`.

### 7.55.2.5 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`

## 7.56 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`

### 7.56.1 Detailed Description

eWIS RDI-L consequent actions

Definition at line 83 of file vtss\_wis\_api.h.

### 7.56.2 Field Documentation

#### 7.56.2.1 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.

#### 7.56.2.2 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.

#### 7.56.2.3 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.

#### 7.56.2.4 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](#)

## 7.57 vtss\_ewis\_sl\_conf\_s Struct Reference

signal label configuration

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

### Data Fields

- `u8 exsl`

#### 7.57.1 Detailed Description

signal label configuration

Definition at line 306 of file vtss\_wis\_api.h.

#### 7.57.2 Field Documentation

##### 7.57.2.1 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](#)

## 7.58 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`

### 7.58.1 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.

### 7.58.2 Field Documentation

#### 7.58.2.1 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.

#### 7.58.2.2 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.

### 7.58.2.3 `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`.

### 7.58.2.4 `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`.

### 7.58.2.5 `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`.

### 7.58.2.6 `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`.

### 7.58.2.7 `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`.

**7.58.2.8 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.

**7.58.2.9 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.

**7.58.2.10 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.

**7.58.2.11 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.

**7.58.2.12 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.

#### 7.58.2.13 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`.

#### 7.58.2.14 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`.

#### 7.58.2.15 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`

## 7.59 vtss\_ewis\_status\_s Struct Reference

eWIS status

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

### Data Fields

- `BOOL fault`
- `BOOL link_stat`

#### 7.59.1 Detailed Description

eWIS status

Definition at line 147 of file `vtss_wis_api.h`.

## 7.59.2 Field Documentation

### 7.59.2.1 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.

### 7.59.2.2 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](#)

## 7.60 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

### 7.60.1 Detailed Description

eWIS test configuration

Definition at line 206 of file vtss\_wis\_api.h.

## 7.60.2 Field Documentation

### 7.60.2.1 loopback

`BOOL vtss_ewis_test_conf_s::loopback`

loop output from Tx to Rx

Definition at line 207 of file `vtss_wis_api.h`.

### 7.60.2.2 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`.

### 7.60.2.3 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`

## 7.61 vtss\_ewis\_test\_status\_s Struct Reference

eWIS test status

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

### Data Fields

- `u16 tstpat_cnt`
- `BOOL ana_sync`

### 7.61.1 Detailed Description

eWIS test status

Definition at line 213 of file vtss\_wis\_api.h.

### 7.61.2 Field Documentation

#### 7.61.2.1 tstamp\_cnt

```
u16 vtss_ewis_test_status_s::tstamp_cnt
```

PRBS31 test pattern error counter.

Definition at line 214 of file vtss\_wis\_api.h.

#### 7.61.2.2 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](#)

## 7.62 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](#)

### 7.62.1 Detailed Description

Trail Trace Identifier type.

Definition at line 55 of file vtss\_wis\_api.h.

### 7.62.2 Field Documentation

#### 7.62.2.1 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.

#### 7.62.2.2 tti

`u8 vtss_ewis_tti_s::tti[64]`

trace identifier value

Definition at line 57 of file vtss\_wis\_api.h.

#### 7.62.2.3 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](#)

## 7.63 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`

### 7.63.1 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.

### 7.63.2 Field Documentation

#### 7.63.2.1 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.

#### 7.63.2.2 tx\_e1

```
u8 vtss_ewis_tx_oh_s::tx_e1
```

Orderwire

Definition at line 227 of file vtss\_wis\_api.h.

### 7.63.2.3 tx\_f1

`u8 vtss_ewis_tx_oh_s::tx_f1`

Section User Channel

Definition at line 228 of file vtss\_wis\_api.h.

### 7.63.2.4 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.

### 7.63.2.5 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.

### 7.63.2.6 tx\_e2

`u8 vtss_ewis_tx_oh_s::tx_e2`

Orderwire

Definition at line 232 of file vtss\_wis\_api.h.

### 7.63.2.7 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.

### 7.63.2.8 tx\_s1

`u8 vtss_ewis_tx_oh_s::tx_s1`

Synchronization messaging

Definition at line 234 of file vtss\_wis\_api.h.

### 7.63.2.9 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.

### 7.63.2.10 tx\_c2

`u8 vtss_ewis_tx_oh_s::tx_c2`

Transmitted C2 path label

Definition at line 237 of file vtss\_wis\_api.h.

### 7.63.2.11 tx\_f2

`u8 vtss_ewis_tx_oh_s::tx_f2`

Path User Channel

Definition at line 238 of file vtss\_wis\_api.h.

### 7.63.2.12 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.

### 7.63.2.13 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](#)

## 7.64 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](#)

### 7.64.1 Detailed Description

eWIS overhead passthru configuration.

Definition at line 245 of file vtss\_wis\_api.h.

### 7.64.2 Field Documentation

#### 7.64.2.1 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.

#### 7.64.2.2 tx\_z0

`BOOL vtss_ewis_tx_passthru_s::tx_z0`

z0 Section growth passthrough

Definition at line 248 of file vtss\_wis\_api.h.

#### 7.64.2.3 tx\_b1

`BOOL vtss_ewis_tx_passthru_s::tx_b1`

b1 BIP passthrough

Definition at line 249 of file vtss\_wis\_api.h.

#### 7.64.2.4 tx\_e1

`BOOL vtss_ewis_tx_passthru_s::tx_e1`

e1 order wire passthrough

Definition at line 250 of file vtss\_wis\_api.h.

#### 7.64.2.5 tx\_f1

`BOOL vtss_ewis_tx_passthru_s::tx_f1`

f1 Section user channel

Definition at line 251 of file vtss\_wis\_api.h.

#### 7.64.2.6 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`.

#### 7.64.2.7 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`.

#### 7.64.2.8 tx\_b2

`BOOL vtss_ewis_tx_passthru_s::tx_b2`

b2 BIP passthrough

Definition at line 256 of file `vtss_wis_api.h`.

#### 7.64.2.9 tx\_k1

`BOOL vtss_ewis_tx_passthru_s::tx_k1`

k1 passthrough

Definition at line 257 of file `vtss_wis_api.h`.

#### 7.64.2.10 tx\_k2

`BOOL vtss_ewis_tx_passthru_s::tx_k2`

k2 passthrough

Definition at line 258 of file `vtss_wis_api.h`.

#### 7.64.2.11 tx\_reil

`BOOL vtss_ewis_tx_passthru_s::tx_reil`

reil passthrough

Definition at line 259 of file vtss\_wis\_api.h.

#### 7.64.2.12 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.

#### 7.64.2.13 tx\_s1

`BOOL vtss_ewis_tx_passthru_s::tx_s1`

Synchronization messaging passthrough

Definition at line 261 of file vtss\_wis\_api.h.

#### 7.64.2.14 tx\_e2

`BOOL vtss_ewis_tx_passthru_s::tx_e2`

order wire passthrough

Definition at line 262 of file vtss\_wis\_api.h.

#### 7.64.2.15 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.

#### 7.64.2.16 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`

## 7.65 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`

#### 7.65.1 Detailed Description

Fan specifications.

Definition at line 1148 of file `vtss_misc_api.h`.

#### 7.65.2 Field Documentation

##### 7.65.2.1 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`.

### 7.65.2.2 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.

### 7.65.2.3 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.

### 7.65.2.4 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.

### 7.65.2.5 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](#)

## 7.66 vtss\_fdma\_cfg\_t Struct Reference

FDMA configuration structure.

```
#include <vtss_fdma_api.h>
```

## Data Fields

- `BOOL enable`
- `u32 rx_mtu`
- `u32 rx_buf_cnt`
- `void(* rx_alloc_cb )(u32 sz, vtss_fdma_list_t *const list, u32 mem_flags)`
- `BOOL rx_dont_strip_vlan_tag`
- `BOOL rx_dont_reinsert_vlan_tag`
- `BOOL rx_allow_vlan_tag_mismatch`
- `BOOL rx_allow_multiple_dcbs`
- `vtss_fdma_list_t *(* rx_cb )(void *ctxt, vtss_fdma_list_t *list)`
- `u32 tx_buf_cnt`
- `void(* tx_done_cb )(void *ctxt, struct tag_vtss_fdma_list *list, vtss_rc rc)`
- `u32 afi_buf_cnt`
- `void(* afi_done_cb )(void *ctxt, struct tag_vtss_fdma_list *list, vtss_rc rc)`

### 7.66.1 Detailed Description

FDMA configuration structure.

The following structure defines the parameters needed to configure the FDMA in general.

Note: For future compatibility, all applications must memset() this structure to 0 before filling it in.

Definition at line 1821 of file vtss\_fdma\_api.h.

### 7.66.2 Field Documentation

#### 7.66.2.1 enable

```
BOOL vtss_fdma_cfg_t::enable
```

Enable FDMA driver.

The FDMA driver can be started and stopped with the use of this member.

If the FDMA driver has once been enabled, it can be paused by setting `enable` to FALSE.

Definition at line 1831 of file vtss\_fdma\_api.h.

### 7.66.2.2 rx\_mtu

`u32 vtss_fdma_cfg_t::rx_mtu`

Rx MTU. The maximum frame length (including FCS) the FDMA will pass on to the application. Typically, an application will set this to  $1518 + (4 * \text{max number of VLAN tags})$ .

Well, in fact, it indicates the number of frame data bytes associated with one DCB. See also [rx\\_allow\\_multiple\\_dcbs](#).

It can be set to 0 to free all memory allocated by the FDMA driver, but if set to a non-zero value, it must be at or greater than 64.

Definition at line 1844 of file vtss\_fdma\_api.h.

### 7.66.2.3 rx\_buf\_cnt

`u32 vtss_fdma_cfg_t::rx_buf_cnt`

Rx buffer count. The number of Rx buffers to allocate. Since the FDMA consists of a number of one or more Rx channels, the allocated buffers will be spread evenly across the Rx channels required on a given platform.

Definition at line 1853 of file vtss\_fdma\_api.h.

### 7.66.2.4 rx\_alloc\_cb

`void(* vtss_fdma_cfg_t::rx_alloc_cb)(u32 sz, vtss_fdma_list_t *const list, u32 mem_flags)`

The FDMA driver allocates its own software DCBs, to obtain correct alignment of the associated hardware DCBs. It uses [VTSS\\_OS\\_MALLOC\(\)](#) for this. DCBs are therefore owned by the FDMA driver.

However, the associated data area, where frame data gets received into, can either be owned by the FDMA driver or the application.

If the application wishes to manage the frame data memory, it must set [rx\\_alloc\\_cb\(\)](#) to a non-NULL value. If the application leaves all the frame data memory management to the FDMA driver, it must set [rx\\_alloc\\_cb\(\)](#) to NULL.

During initialization, the FDMA driver allocates [rx\\_buf\\_cnt](#) DCBs and checks whether the application will allocate the corresponding frame data, or it should do that itself. If [rx\\_alloc\\_cb\(\)](#) is NULL, the FDMA driver will call [VTSS\\_OS\\_MALLOC\(\)](#) to allocate the corresponding frame data. Otherwise it will call [rx\\_alloc\\_cb\(\)](#) to get it allocated.

DCBs are the glue between the FDMA driver and the application, in the sense that once the FDMA has received a frame, it passes a pointer to the DCB to the application's [rx\\_cb\(\)](#) callback. At this point the application has three options: 1) Pass the DCB further up the food chain to higher parts of the application, and once it has been handled, feed it back to the FDMA driver with a call to [vtss\\_fdma\\_dcb\\_release\(\)](#). In this case, the [rx\\_cb\(\)](#) function must return NULL. 2) Detach the frame data from the DCB, set the DCB's alloc\_ptr to NULL, and return the DCB back to the FDMA driver through the return value of [rx\\_cb\(\)](#). 3) If - for some reason - the application chooses not to pass the frame further up the food chain in the call to [rx\\_cb\(\)](#), it may simply return the DCB as is from [rx\\_cb\(\)](#), without altering the alloc\_ptr member.

Whether or not the DCB is returned directly as a return value from [rx\\_cb\(\)](#) or returned through a call to [vtss\\_fdma\\_dcb\\_release\(\)](#), the alloc\_ptr member of the DCB indicates whether the frame data has been absorbed by the application or not, as follows: A) If alloc\_ptr == NULL when the DCB comes back, the FDMA assumes that the frame data has been absorbed by the application. In that case, it will ask the application to re-allocate the alloc\_ptr member by calling [rx\\_alloc\\_cb\(\)](#) (which must therefore be non-NULL). B) If alloc\_ptr is non-NULL when the DCB comes back to the FDMA driver, the FDMA driver will just re-initialize the DCB and not call the [rx\\_alloc\\_cb\(\)](#) function.

[rx\\_alloc\\_cb\(\)](#) takes three arguments ([IN] params are seen from the application's point of view).

### Parameters

<i>sz</i>	[IN] Number of bytes to allocate.
<i>list</i>	[INOUT] A NULL-terminated list of DCBs to get allocated frame data for. It may consist of more than one DCB if the application has chosen to support reception of frames fragmented over multiple DCBs ( <a href="#">rx_allow_multiple_dcbs</a> ) and during initialization. The application must fill in the alloc_ptr and possibly also the "user" members of the DCB.
<i>mem_flags</i>	[IN] This is a bitwise OR of the vtss_mem_flags_t, enumeration, which tells the application how to allocate the memory.

If - upon return from the [rx\\_alloc\\_cb\(\)](#) function, the DCB's alloc\_ptr member is still NULL, one of two things will happen: i) If it happens during initialization ([vtss\\_fdma\\_cfg\(\)](#)), the function call will fail, and the FDMA will not be started. ii) If it happens at runtime ([rx\\_cb\(\)](#)/[vtss\\_fdma\\_dcb\\_release\(\)](#)), the DCB will be put in a special list that can only be released if the FDMA driver gets restarted.

The [rx\\_alloc\\_cb\(\)](#) will be invoked for every DCB it should supply frame data for, so don't use the ->next member.

Definition at line 1926 of file vtss\_fdma\_api.h.

### 7.66.2.5 rx\_dont\_strip\_vlan\_tag

`BOOL vtss_fdma_cfg_t::rx_dont_strip_vlan_tag`

Don't strip VLAN tag.

The switch hardware does not strip VLAN tags from the frames prior to received by software.

The FDMA driver can be configured to strip VLAN tags from frames received on ports where the VLAN tag matches the port's VLAN tag setup. This is useful for, e.g. IP stacks that don't expect VLAN tags inside the frames. The recommendation is to set this to FALSE.

Definition at line 1939 of file vtss\_fdma\_api.h.

### 7.66.2.6 rx\_dont\_reinsert\_vlan\_tag

`BOOL vtss_fdma_cfg_t::rx_dont_reinsert_vlan_tag`

Don't re-insert VLAN tag.

This is obsolete and is ignored by the FDMA code.

Definition at line 1946 of file vtss\_fdma\_api.h.

### 7.66.2.7 rx\_allow\_vlan\_tag\_mismatch

`BOOL vtss_fdma_cfg_t::rx_allow_vlan_tag_mismatch`

If a frame is received with a VLAN tag TPID that does not match the port's setup, it can be dropped (DCB gets recycled) by the FDMA driver, by setting `rx_allow_vlan_tag_mismatch` to FALSE.

Definition at line 1953 of file vtss\_fdma\_api.h.

### 7.66.2.8 rx\_allow\_multiple\_dcbs

`BOOL vtss_fdma_cfg_t::rx_allow_multiple_dcbs`

Rather than indicating the maximum frame size, `rx_mtu` indicates the maximum number of frame data bytes that can go into one DCB's data area. If the application can handle a frame spanning more than one DCB, it can set `rx_allow_multiple_dcbs` to TRUE, in which case the `rx_cb()` callback function may be invoked with a list of DCBs (the SOF DCB's next pointer is non-NULL).

In this way, the application may support jumbo frames without having to allocate jumbo frame size bytes to each DCB.

Definition at line 1965 of file vtss\_fdma\_api.h.

### 7.66.2.9 rx\_cb

`vtss_fdma_list_t*(* vtss_fdma_cfg_t::rx_cb)(void *ctxt, vtss_fdma_list_t *list)`

The address of the callback function invoked by the FDMA driver code when a frame arrives. The callback function is invoked once per frame.

The function is invoked in the same context as the application invoked `vtss_fdma_irq_handler()`.

The parameters to the callback function are as follows:

1. ctxt: The ctxt passed to `vtss_fdma_irq_handler()`.
2. list: NULL-terminated list of software DCBs making up the frame. Depending on the frame size one or more DCBs are used. The last is always NULL-terminated. The FDMA has filled in the following fields (see definition of `vtss_fdma_list_t` structure for details): (@dcb), (@\*data, @act\_len, (@next)). Once delivered to the Packet Module, the FDMA will not use the list again. The FDMA can, however, be handed back a (new) list of software DCBs by returning the list when the `rx_cb()` returns. Alternatively, use the `vtss_fdma_dcb_release()` function.
  - Expected return value from `rx_cb()`:  
See discussion under `rx_alloc_cb`.

Definition at line 1988 of file vtss\_fdma\_api.h.

### 7.66.2.10 tx\_buf\_cnt

`u32 vtss_fdma_cfg_t::tx_buf_cnt`

Tx buffer count. The number of Tx DCBs to allocate.

Definition at line 1994 of file `vtss_fdma_api.h`.

### 7.66.2.11 tx\_done\_cb

`void(* vtss_fdma_cfg_t::tx_done_cb) (void *cntxt, struct tag_vtss_fdma_list *list, vtss_rc rc)`

The address of the callback function invoked by the FDMA driver code when a frame has been transmitted. The callback function is invoked once per frame.

The function is invoked in the same context as the application invoked `vtss_fdma_irq_handler()`.

The parameters to the callback function are as follows:

1. cntxt: The cntxt passed to `vtss_fdma_irq_handler()`.
2. list: NULL-terminated list of software DCBs making up the frame. The last is always NULL-terminated.
3. rc: Return code. VTSS\_RC\_OK on success, otherwise failed to transmit.

The application must extract the required information from the DCB list (e.g. through the DCB's user field) prior to returning from this function, because the FDMA driver will recycle the DCBs upon return.

In the event that the DCBs were originally coming from extraction, the FDMA driver returns the DCBs back to extraction.

Definition at line 2016 of file `vtss_fdma_api.h`.

### 7.66.2.12 afi\_buf\_cnt

`u32 vtss_fdma_cfg_t::afi_buf_cnt`

AFI buffer count. The number of AFI DCBs to allocate.

Definition at line 2023 of file `vtss_fdma_api.h`.

### 7.66.2.13 afi\_done\_cb

```
void(* vtss_fdma_cfg_t::afi_done_cb) (void *cntxt, struct tag_vtss_fdma_list *list, vtss_rc rc)
```

The address of the callback function invoked by the FDMA driver code when an AFI frame has been cancelled. The callback function is invoked once per frame.

The function is invoked in the same context as the application invoked [vtss\\_fdma\\_irq\\_handler\(\)](#).

The parameters to the callback function are as follows:

1. ctxt: The ctxt passed to [vtss\\_fdma\\_irq\\_handler\(\)](#).
2. list: NULL-terminated list of software DCBs making up the frame. The last is always NULL-terminated.
3. rc: Return code. VTSS\_RC\_OK always.

The application must extract the required information from the DCB list (e.g. through the DCB's user field) prior to returning from this function, because the FDMA driver will recycle the DCBs upon return.

Definition at line 2043 of file vtss\_fdma\_api.h.

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

- vtss\_api/include/[vtss\\_fdma\\_api.h](#)

## 7.67 vtss\_fdma\_ch\_cfg\_t Struct Reference

Channel configuration structure.

```
#include <vtss_fdma_api.h>
```

### Data Fields

- [vtss\\_fdma\\_ch\\_usage\\_t usage](#)
- [vtss\\_packet\\_rx\\_grp\\_t xtr\\_grp](#)
- [u32 inj\\_grp\\_mask](#)
- [vtss\\_fdma\\_list\\_t \\* list](#)
- [vtss\\_fdma\\_list\\_t \\*\(\\* xtr\\_cb \)\(void \\*ctxt, vtss\\_fdma\\_list\\_t \\*list, vtss\\_packet\\_rx\\_queue\\_t qu\)](#)
- [int prio](#)
- [vtss\\_chip\\_no\\_t chip\\_no](#)
- [u32 ccm\\_quotient\\_max](#)

### 7.67.1 Detailed Description

Channel configuration structure.

The following structure defines the parameters needed to configure a DMA channel. The DMA channel can either be used for frame extraction, frame injection, or period frame injection. This is controlled by the [usage](#) parameter.

The interpretation and validity of the remaining fields depend on the [usage](#) parameter.

Note: For future compatibility, all applications must memset() this structure to 0 before filling it in.

Definition at line 1596 of file vtss\_fdma\_api.h.

## 7.67.2 Field Documentation

### 7.67.2.1 usage

```
vtss_fdma_ch_usage_t vtss_fdma_ch_cfg_t::usage
```

#### XTR/INJ/AFI:

Indicates whether this channel is used for extraction, injection, or periodic injection. Luton26 and Jaguar note: At most one channel may be configured for AFI use. To disable a channel, set this member to VTSS\_FDMA\_CH\_USAGE\_UNUSED.

Definition at line 1604 of file vtss\_fdma\_api.h.

### 7.67.2.2 xtr\_grp

```
vtss_packet_rx_grp_t vtss_fdma_ch_cfg_t::xtr_grp
```

#### XTR:

The extraction group that this channel serves. Need only be set if cfg->chip\_no == 1. Valid values are [0; VTSS\_PACKET\_RX\_GRP\_CNT[

#### INJ/AFI:

Unused.

Validity: Jaguar1: Y - for 48-port, only. Luton26: N Serval : N Jaguar2: N Serval2: N ServalT: N

Definition at line 1624 of file vtss\_fdma\_api.h.

### 7.67.2.3 inj\_grp\_mask

```
u32 vtss_fdma_ch_cfg_t::inj_grp_mask
```

#### XTR:

Unused.

#### INJ:

A mask containing the injection groups that this channel serves. A channel may serve more than one injection group. On some architectures a given injection group serves a given type of service (e.g. super- priority injection, switched injection, front-port directed injection, etc.). On such architectures, the actual injection group to use is conveyed in the call to vtss\_fdma\_inj().

At least one bit must be set in the mask. The most significant bit that can be set is VTSS\_PACKET\_TX\_GRP\_CNT - 1.

#### AFI:

A mask containing the injection group that the AFI channel serves. This is a one-hot mask, that is, exactly one bit must be set, and the most significant bit that can be set is VTSS\_PACKET\_TX\_GRP\_CNT - 1.

Validity: Jaguar : Y Luton26: Y, one-hot (exactly one bit set) Serval : Not used. Implicitly set through the channel number. Jaguar2: N Serval2: N ServalT: N

Definition at line 1654 of file vtss\_fdma\_api.h.

## 7.67.2.4 list

```
vtss_fdma_list_t* vtss_fdma_ch_cfg_t::list
```

**XTR:**

A linked, NULL-terminated list of software DCBs, into which the FDMA extract frames. One frame may take one or more DCBs, depending on the size of the associated data area. The following fields of each list item must be pre-initialized by the Packet Module (see definition of vtss\_fdma\_list\_t structure for details): @data, @alloc\_len, (user), and @next.

**INJ:**

Unused.

**AFI:**

Unused. Must be NULL. The FDMA driver allocates DCBs on its own. This approach is chosen because it's very hard to pre-determine a good number of DCBs that must be allocated, since it depends on whether the application uses frame counting or sequence numbering or not, and whether it may add or cancel frames so fast that both a running, a pending, and a new list of frames must be built up.

Definition at line 1677 of file vtss\_fdma\_api.h.

## 7.67.2.5 xtr\_cb

```
vtss_fdma_list_t*(* vtss_fdma_ch_cfg_t::xtr_cb) (void *cntxt, vtss_fdma_list_t *list, vtss_packet_rx_queue_t qu)
```

**XTR:**

The address of the callback function invoked by the FDMA driver code when a frame arrives. The callback function is invoked once per frame.

- Call context:  
DSR (the same as what invokes [vtss\\_fdma\\_irq\\_handler\(\)](#)).
- The parameters to the callback function are as follows:
  1. ctxt: The value passed in the Packet Module's call to [vtss\\_fdma\\_irq\\_handler\(\)](#).
  2. list: NULL-terminated list of software DCBs making up the frame. Depending on the frame size one or more DCBs are used. The last is always NULL-terminated. The FDMA has filled in the following fields (see definition of vtss\_fdma\_list\_t structure for details): (@dcb), (@\*data, @act\_len, (@next)). Once delivered to the Packet Module, the FDMA will not use the list again. The FDMA can, however, be handed back a (new) list of software DCBs by returning the list when the [xtr\\_cb\(\)](#) returns. Alternatively, use the [vtss\\_fdma\\_xtr\\_add\\_dcbs\(\)](#) function.
  3. qu: The extraction queue that this frame was extracted from. This may be useful if all extraction callback functions map to the same function.
    - Expected return value from [xtr\\_cb\(\)](#):  
Non-NULL if the Packet Module wishes to give the FDMA new DCBs to extract to. In the case where the Packet Module handles the frame directly, it could as well pass back the list that [xtr\\_cb\(\)](#) was called with right away. In the case where the frame is passed up to higher levels, like an IP stack, the Packet Module will probably return NULL in the call to [xtr\\_cb\(\)](#) and provide a replacement list at a later point in time with a call to [vtss\\_fdma\\_xtr\\_add\\_dcbs\(\)](#).

**INJ/AFI:**

Unused.

Definition at line 1710 of file vtss\_fdma\_api.h.

### 7.67.2.6 prio

```
int vtss_fdma_ch_cfg_t::prio
```

#### XTR/INJ/AFI:

Controls the channel priority. Valid values are [0; 7]. The higher value the higher priority. Everytime the DMA H/W needs to find the next channel to serve, it selects - amongst the pending channels - the channel with the highest priority. Two or more channels may have the same priority, in which case the DMA H/W grants the lowest-numbered channel access. Note: If using the deprecated vtss\_fdma\_inj\_cfg() and vtss\_fdma\_xtr\_cfg() functions, the channel priority will be the same as the channel number.

Definition at line 1725 of file vtss\_fdma\_api.h.

### 7.67.2.7 chip\_no

```
vtss_chip_no_t vtss_fdma_ch_cfg_t::chip_no
```

#### XTR

In a dual-chip solution, this controls the chip to extract from. For single chip solutions, this field must be set to 0. For dual-chip solutions, this field must be set to 0 for extraction from the primary chip, and in this case, the xtr\_grp must match the channel number. For dual-chip solutions, this field must be set to 1 for extraction from the secondary chip, and in this case, the xtr\_grp need not match the channel number, because - at the time of writing - the channel number isn't really used for anything in that case, because the secondary chip's frames have to be read out manually because they can't be auto-transferred to the primary chip's CPU extraction queues without losing the CPU Rx queue number that it was stored in on the secondary chip.

#### INJ/AFI:

Unused.

Definition at line 1746 of file vtss\_fdma\_api.h.

### 7.67.2.8 ccm\_quotient\_max

```
u32 vtss_fdma_ch_cfg_t::ccm_quotient_max
```

#### AFI:

Controls the channel's maximum frequency quotient. A given AFI channel can handle several frame frequencies provided they are multiples of each other. This configuration option determines the maximum frequency quotient two frames running on the same channel can have. If for instance, it's set to 2 and the first frame is transmitted with 50 fps, then a subsequent frame can be transmitted with a requested frequency of 25, 50, or 100 fps, only. To allow only one frequency on the channel, set it to 1.

Valid values are in the range [1; VTSS\_FDMA\_CCM\_QUOTIENT\_MAX].

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

Definition at line 1776 of file vtss\_fdma\_api.h.

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

- vtss\_api/include/vtss\_fdma\_api.h

## 7.68 vtss\_fdma\_throttle\_cfg\_t Struct Reference

```
#include <vtss_fdma_api.h>
```

### Data Fields

- `u32 frm_limit_per_tick [VTSS_PACKET_RX_QUEUE_CNT]`
- `u32 byte_limit_per_tick [VTSS_PACKET_RX_QUEUE_CNT]`
- `u32 suspend_tick_cnt [VTSS_PACKET_RX_QUEUE_CNT]`

#### 7.68.1 Detailed Description

In order to survive e.g. broadcast storms, the FDMA driver incorporates a poor man's policing/throttling scheme.

The idea is to check upon every frame reception whether the CPU Rx queue on which the frame was received has exceeded its limit, and if so, suspend extraction from the queue for a period of time.

The FDMA has no notion of time, so this requires a little help from the application. To take advantage of the feature, the application must first call `vtss_fdma_throttle_cfg_set()` with an appropriate configuration, and then call `vtss_fdma_throttle_tick()` on a regular, application-defined basis, e.g. 10 times per second.

The throttle tick takes care of re-opening the queue after the suspension period elapses.

Notice that once an extraction queue gets disabled, that extraction queue will no longer be a source of interrupts. The feature will only affect extraction queues for which it is enabled.

Once disabling an extraction queue, the remaining frames in the queue will be read out, after which the queue will be silent. When re-enabling, it will be fresh frames that come in.

Be aware that on Lu26, the trick to disable a queue involves directing the frames to the second CPU port (physical #27). If your application uses two CPU ports, then throttling will have unexpected side-effects.

There is no need to call `vtss_fdma_throttle_tick()` unless throttling is enabled for at least one extraction queue.

If throttling is enabled for at least one queue, and `vtss_fdma_throttle_tick()` is *not* called, you risk that an extraction queue will get disabled and never re-enabled again.

Validity: Luton26: Y Jaguar1: Y Serval1: Y Jaguar2: Y Serval2: Y ServalT: Y

Definition at line 2205 of file vtss\_fdma\_api.h.

#### 7.68.2 Field Documentation

##### 7.68.2.1 frm\_limit\_per\_tick

```
u32 vtss_fdma_throttle_cfg_t::frm_limit_per_tick [VTSS_PACKET_RX_QUEUE_CNT]
```

Controls - per extraction queue - the maximum number of frames extracted between two calls to `vtss_fdma_throttle_tick()` without suspending extraction from that queue.

If 0, frame count throttling is disabled for that extraction queue.

Definition at line 2225 of file vtss\_fdma\_api.h.

### 7.68.2.2 byte\_limit\_per\_tick

```
u32 vtss_fdma_throttle_cfg_t::byte_limit_per_tick[VTSS_PACKET_RX_QUEUE_CNT]
```

Controls - per extraction queue - the maximum number of bytes extracted between two calls to [vtss\\_fdma\\_throttle\\_tick\(\)](#) without suspending extraction from that queue.

If 0, byte count throttling is disabled for that extraction queue.

Definition at line 2236 of file vtss\_fdma\_api.h.

### 7.68.2.3 suspend\_tick\_cnt

```
u32 vtss_fdma_throttle_cfg_t::suspend_tick_cnt[VTSS_PACKET_RX_QUEUE_CNT]
```

Controls - per extraction queue - the number of invocations of [vtss\\_fdma\\_throttle\\_tick\(\)](#) that must happen before an extraction queue that has been disabled, gets re-enabled.

For instance, a value of 0 means: re-enable the extraction queue on the next tick. a value of 1 means: re-enable the extraction queue two ticks from when it was suspended.

Definition at line 2248 of file vtss\_fdma\_api.h.

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

- vtss\_api/include/[vtss\\_fdma\\_api.h](#)

## 7.69 vtss\_fdma\_tx\_info\_t Struct Reference

FDMA Injection Properties.

```
#include <vtss_fdma_api.h>
```

### Data Fields

- void \* [pre\\_cb\\_ctxt1](#)
- void \* [pre\\_cb\\_ctxt2](#)
- void(\* [pre\\_cb](#) )(void \*ctxt1, void \*ctxt2, [vtss\\_fdma\\_list\\_t](#) \*list)
- u32 [afi\\_fps](#)
- [vtss\\_fdma\\_afi\\_type\\_t](#) [afi\\_type](#)
- BOOL [afi\\_enable\\_counting](#)
- BOOL [afi\\_enable\\_sequence\\_numbering](#)
- u16 [afi\\_sequence\\_number\\_offset](#)

### 7.69.1 Detailed Description

FDMA Injection Properties.

Definition at line 743 of file vtss\_fdma\_api.h.

## 7.69.2 Field Documentation

### 7.69.2.1 pre\_cb\_ctxt1

```
void* vtss_fdma_tx_info_t::pre_cb_ctxt1
```

Any user data. Is passed in a call to [@pre\\_cb\(\)](#).

Definition at line 747 of file vtss\_fdma\_api.h.

### 7.69.2.2 pre\_cb\_ctxt2

```
void* vtss_fdma_tx_info_t::pre_cb_ctxt2
```

Any user data. Is paased in a call to [@pre\\_cb\(\)](#).

Definition at line 752 of file vtss\_fdma\_api.h.

### 7.69.2.3 pre\_cb

```
void(* vtss_fdma_tx_info_t::pre_cb) (void *ctxt1, void *ctxt2, vtss\_fdma\_list\_t *list)
```

Callback function called just before the frame is handed over to the FDMA H/W. If NULL, no callback will occur. The called back function may change frame data, nothing else. The called back function *MUST* execute fast and without waiting for synchronization primitives, i.e. no waits are allowed. When called back, interrupts may be disabled. Not used for non-SOF items, since the callback is only called once per frame.

Implementation notes: Due to shuffling of beginning-of-frame-fields, this will not work for the following architectures under the specified circumstances: Luton26: This will not work for switched frames. Serval : This will not work for switched frames.

Parameters:

- ctxt1: The value of [@pre\\_cb\\_ctxt1](#).
- ctxt2: The value of [@pre\\_cb\\_ctxt2](#).
- list: Pointer to the SOF item. Validity: Luton26: Y Jaguar1: Y Serval : Y Jaguar2: Y Serval2: Y ServalT: Y

Definition at line 780 of file vtss\_fdma\_api.h.

#### 7.69.2.4 afi\_fps

`u32 vtss_fdma_tx_info_t::afi_fps`

Number of times this frame will be injected per second.

There is a number of restrictions on AFI frames:

- `pre_cb` must be NULL,
- `vtss_packet_tx_info_t::switch_frm` must be FALSE,
- `vtss_packet_tx_info_t::dst_port_mask` can at most have one bit set, i.e. the frame cannot be multicast,
- `vtss_packet_tx_info_t::cos` must not be 8 (i.e. super-prio not supported),
- `vtss_packet_tx_info_t::tx_vstax_hdr` must be VTSS\_PACKET\_TX\_VSTAX\_NONE,
- The frame must be held in one single DCB.

The memory that contains the actual frame that `vtss_fdma_tx()` is called with will not be releasable until the AFI frame is cancelled (see `vtss_fdma_afi_cancel()`).

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

Definition at line 818 of file `vtss_fdma_api.h`.

#### 7.69.2.5 afi\_type

`vtss_fdma_afi_type_t vtss_fdma_tx_info_t::afi_type`

If the platform supports both FDMA-based and switch-core-based AFI, the application must have a way to select one or the other when requesting injection of an AFI frame.

Currently any given platform only supports one or the other, so there's no reason to set it to anything but VTSS\_←FDMA\_AFI\_TYPE\_AUTO.

Definition at line 830 of file `vtss_fdma_api.h`.

#### 7.69.2.6 afi\_enable\_counting

`BOOL vtss_fdma_tx_info_t::afi_enable_counting`

Indicates whether this frame is subject to counting.

It is only used when `afi_fps` is > 0.

The number of frames injected is returned in the call to `tx_done_cb()` once the frame is cancelled. Interim counters can be obtained through `vtss_fdma_list_t::afi_frm_cnt`.

Enabling counting has some side effects, because the counting must be done in S/W. First of all, to avoid overloading the CPU with interrupts, the FDMA driver makes sure that the frame is repeated a number of times to only get interrupts every, say, 50 ms. This requires an awful lot of DCBs, which will be dynamically allocated by the FDMA driver.

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

Definition at line 857 of file `vtss_fdma_api.h`.

### 7.69.2.7 afi\_enable\_sequence\_numbering

`BOOL vtss_fdma_tx_info_t::afi_enable_sequence_numbering`

Indicates whether this frame is subject to sequence number updating.

It is only used when `afi_fps` is > 0.

The offset within the frame to update is set with `afi_sequence_number_offset`.

Enabling sequence numbering has some side effects, because the frame updates must be done in S/W. First of all, to avoid overloading the CPU with interrupts, the frame will be copied a number of times to only get interrupts every, say, 50 ms. This requires dynamic memory allocation within the FDMA driver. Let's take an example: Suppose you wish to send 64-byte frames @ 200 Mbps and have these frames sequence numbered. Such frames will give  $200,000,000 / (8 * 64) \approx 400,000$  frames per second. In other words, the temporal spacing between these frames is 2.5 microseconds, which - to get to a 50 ms interrupt rate - means that the frames must be copied  $50000 / 2.5 = 20000$  times. With overhead, one 64-byte frame actually requires 88 bytes, so the total amount of (dynamic) memory required for this operation is 1.76 MBytes (not taking into account the DCBs, which are also dynamically allocated within the driver). In fact, you have to double this number, because the driver is made in such a way that the H/W is not stopped while updating the sequence numbers. Instead, the FDMA driver makes a circular list consisting of two 50 ms half-circles, so that S/W can update one half while the H/W is injecting the other half.

SO BE CAREFUL WHEN YOU DESIGN THE APPLICATION!

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

Definition at line 897 of file `vtss_fdma_api.h`.

### 7.69.2.8 afi\_sequence\_number\_offset

`u16 vtss_fdma_tx_info_t::afi_sequence_number_offset`

This field indicates the zero-based byte-offset within the frame to increment by one.

It is only used when `afi_enable_sequence_numbering` is TRUE.

The initial value of the sequence number is whatever the frame you inject contains at that offset.

The FDMA driver assumes a 4-byte, any-aligned, network-ordered (big endian) value, which wraps to 0 at `0xFFFF←FFFF`.

Needless to say that the full 4-byte value must be contained within the frame's length.

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

Definition at line 922 of file `vtss_fdma_api.h`.

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

- `vtss_api/include/vtss_fdma_api.h`

## 7.70 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`
- `vtss_gpio_10g_led_conf_t led_conf`
- `BOOL invert_output`

#### 7.70.1 Detailed Description

GPIO configured mode.

Definition at line 2513 of file vtss\_phy\_10g\_api.h.

#### 7.70.2 Field Documentation

##### 7.70.2.1 mode

```
vtss_10g_phy_gpio_t vtss_gpio_10g_gpio_mode_t::mode
```

Mode of this GPIO pin

Definition at line 2515 of file vtss\_phy\_10g\_api.h.

##### 7.70.2.2 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\_RXFAULT mode, this is the PCS status port number that is related to this GPIO

Definition at line 2516 of file vtss\_phy\_10g\_api.h.

### 7.70.2.3 input

`vtss_gpio_10g_input_t` vtss\_gpio\_10g\_gpio\_mode\_t::input

GPIO input modes

Definition at line 2518 of file vtss\_phy\_10g\_api.h.

### 7.70.2.4 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 2519 of file vtss\_phy\_10g\_api.h.

### 7.70.2.5 p\_gpio

`vtss_gpio_10g_no_t` vtss\_gpio\_10g\_gpio\_mode\_t::p\_gpio

Per channel GPIO number

Definition at line 2520 of file vtss\_phy\_10g\_api.h.

### 7.70.2.6 c\_intrpt

`vtss_gpio_10g_chan_intrpt_t` vtss\_gpio\_10g\_gpio\_mode\_t::c\_intrpt

Per Channel interrupt,

Definition at line 2521 of file vtss\_phy\_10g\_api.h.

### 7.70.2.7 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 2522 of file vtss\_phy\_10g\_api.h.

#### 7.70.2.8 aggr\_intrpt

`u32 vtss_gpio_10g_gpio_mode_t::aggr_intrpt`

Bitmask corresponds to aggregated interrupt

Definition at line 2523 of file `vtss_phy_10g_api.h`.

#### 7.70.2.9 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 2524 of file `vtss_phy_10g_api.h`.

#### 7.70.2.10 p\_gpio\_intrpt

`BOOL vtss_gpio_10g_gpio_mode_t::p_gpio_intrpt`

Port GPIO Change interrupt

Definition at line 2525 of file `vtss_phy_10g_api.h`.

#### 7.70.2.11 led\_conf

`vtss_gpio_10g_led_conf_t vtss_gpio_10g_gpio_mode_t::led_conf`

Led mode configuration

Definition at line 2526 of file `vtss_phy_10g_api.h`.

#### 7.70.2.12 invert\_output

`BOOL vtss_gpio_10g_gpio_mode_t::invert_output`

Inverts value on the GPIO's output

Definition at line 2527 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`

## 7.71 vtss\_gpio\_10g\_led\_conf\_t Struct Reference

LED Mode selection.

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

### Data Fields

- `vtss_gpio_10g_led_mode_t mode`
- `vtss_gpio_10g_led_blink_t blink`

#### 7.71.1 Detailed Description

LED Mode selection.

Definition at line 2505 of file vtss\_phy\_10g\_api.h.

#### 7.71.2 Field Documentation

##### 7.71.2.1 blink

```
vtss_gpio_10g_led_blink_t vtss_gpio_10g_led_conf_t::blink
```

Led mode selection

Definition at line 2507 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

## 7.72 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`

### 7.72.1 Detailed Description

Initialization configuration.

Definition at line 515 of file vtss\_init\_api.h.

### 7.72.2 Field Documentation

#### 7.72.2.1 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.

#### 7.72.2.2 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.

### 7.72.2.3 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.

### 7.72.2.4 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.

### 7.72.2.5 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.

### 7.72.2.6 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.

### 7.72.2.7 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.

#### 7.72.2.8 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.

#### 7.72.2.9 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.

#### 7.72.2.10 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.

#### 7.72.2.11 warm\_start\_enable

```
BOOL vtss_init_conf_t::warm_start_enable
```

Allow warm start

Definition at line 535 of file vtss\_init\_api.h.

#### 7.72.2.12 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.

### 7.72.2.13 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.

### 7.72.2.14 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.

### 7.72.2.15 pi

`vtss_pi_conf_t` `vtss_init_conf_t::pi`

Parallel Interface configuration

Definition at line 547 of file vtss\_init\_api.h.

### 7.72.2.16 serdes

`vtss_serdes_macro_conf_t` `vtss_init_conf_t::serdes`

Serdes macro configuration

Definition at line 550 of file vtss\_init\_api.h.

### 7.72.2.17 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](#)

## 7.73 vtss\_inst\_create\_t Struct Reference

Create structure.

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

### Data Fields

- [vtss\\_target\\_type\\_t target](#)

#### 7.73.1 Detailed Description

Create structure.

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

#### 7.73.2 Field Documentation

##### 7.73.2.1 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`

## 7.74 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](#)

### 7.74.1 Detailed Description

Either an IPv4 or IPv6 address.

Definition at line 813 of file types.h.

### 7.74.2 Field Documentation

#### 7.74.2.1 type

`vtss_ip_type_t vtss_ip_addr_t::type`

Union type

Definition at line 814 of file types.h.

#### 7.74.2.2 ipv4

`vtss_ipv4_t vtss_ip_addr_t::ipv4`

IPv4 address

Definition at line 816 of file types.h.

#### 7.74.2.3 ipv6

`vtss_ipv6_t vtss_ip_addr_t::ipv6`

IPv6 address

Definition at line 817 of file types.h.

#### 7.74.2.4 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`

## 7.75 vtss\_ip\_network\_t Struct Reference

IPv6 network.

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

### Data Fields

- `vtss_ip_addr_t address`
- `vtss_prefix_size_t prefix_size`

#### 7.75.1 Detailed Description

IPv6 network.

Definition at line 836 of file types.h.

#### 7.75.2 Field Documentation

##### 7.75.2.1 address

```
vtss_ip_addr_t vtss_ip_network_t::address
```

Network address

Definition at line 838 of file types.h.

##### 7.75.2.2 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`

## 7.76 vtss\_ipv4\_network\_t Struct Reference

IPv4 network.

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

## Data Fields

- [vtss\\_ipv4\\_t address](#)
- [vtss\\_prefix\\_size\\_t prefix\\_size](#)

### 7.76.1 Detailed Description

IPv4 network.

Definition at line 822 of file types.h.

### 7.76.2 Field Documentation

#### 7.76.2.1 address

[vtss\\_ipv4\\_t](#) vtss\_ipv4\_network\_t::address

Network address

Definition at line 824 of file types.h.

#### 7.76.2.2 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](#)

## 7.77 vtss\_ipv4\_uc\_t Struct Reference

IPv4 unicast routing entry.

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

## Data Fields

- [vtss\\_ipv4\\_network\\_t network](#)
- [vtss\\_ipv4\\_t destination](#)

### 7.77.1 Detailed Description

IPv4 unicast routing entry.

Definition at line 852 of file types.h.

### 7.77.2 Field Documentation

#### 7.77.2.1 network

`vtss_ipv4_network_t vtss_ipv4_uc_t::network`

Network to route

Definition at line 854 of file types.h.

#### 7.77.2.2 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`

## 7.78 vtss\_ipv6\_network\_t Struct Reference

IPv6 network.

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

### Data Fields

- `vtss_ipv6_t address`
- `vtss_prefix_size_t prefix_size`

### 7.78.1 Detailed Description

IPv6 network.

Definition at line 829 of file types.h.

### 7.78.2 Field Documentation

#### 7.78.2.1 address

`vtss_ipv6_t vtss_ipv6_network_t::address`

Network address

Definition at line 831 of file types.h.

#### 7.78.2.2 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`

## 7.79 vtss\_ipv6\_t Struct Reference

IPv6 address/mask.

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

### Data Fields

- `u8 addr [16]`

### 7.79.1 Detailed Description

IPv6 address/mask.

Definition at line 797 of file types.h.

## 7.79.2 Field Documentation

### 7.79.2.1 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](#)

## 7.80 vtss\_ipv6\_uc\_t Struct Reference

IPv6 routing entry.

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

### Data Fields

- [vtss\\_ipv6\\_network\\_t network](#)
- [vtss\\_ipv6\\_t destination](#)

### 7.80.1 Detailed Description

IPv6 routing entry.

Definition at line 860 of file types.h.

## 7.80.2 Field Documentation

### 7.80.2.1 network

`vtss_ipv6_network_t vtss_ipv6_uc_t::network`

Network to route

Definition at line 862 of file types.h.

### 7.80.2.2 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`

## 7.81 vtss\_irq\_conf\_t Struct Reference

Interrupt configuration options.

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

### Data Fields

- `BOOL external`
- `u8 destination`

### 7.81.1 Detailed Description

Interrupt configuration options.

Definition at line 972 of file vtss\_misc\_api.h.

### 7.81.2 Field Documentation

#### 7.81.2.1 external

`BOOL vtss_irq_conf_t::external`

Redirect to external IRQ

Definition at line 973 of file vtss\_misc\_api.h.

### 7.81.2.2 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](#)

## 7.82 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](#)

### 7.82.1 Detailed Description

Interrupt status structure.

Definition at line 980 of file vtss\_misc\_api.h.

### 7.82.2 Field Documentation

#### 7.82.2.1 active

```
u32 vtss_irq_status_t::active
```

Bitmap for pending IRQs (VTSS\_IRQ\_xxx)

Definition at line 981 of file vtss\_misc\_api.h.

### 7.82.2.2 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.

### 7.82.2.3 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.

### 7.82.2.4 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

## 7.83 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`

### 7.83.1 Detailed Description

Common configurations for all routing legs.

Definition at line 163 of file vtss\_l3\_api.h.

## 7.83.2 Field Documentation

### 7.83.2.1 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.

### 7.83.2.2 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.

### 7.83.2.3 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`

## 7.84 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`

### 7.84.1 Detailed Description

Routing interface statics counter.

Definition at line 886 of file types.h.

### 7.84.2 Field Documentation

#### 7.84.2.1 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.

#### 7.84.2.2 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.

#### 7.84.2.3 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.

#### 7.84.2.4 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.

#### 7.84.2.5 ipv4uc\_transmitted\_octets

`u64 vtss_l3_counters_t::ipv4uc_transmitted_octets`

IPv4UC octets transmitted

Definition at line 892 of file types.h.

#### 7.84.2.6 ipv4uc\_transmitted\_frames

`u64 vtss_l3_counters_t::ipv4uc_transmitted_frames`

IPv4UC frames transmitted

Definition at line 893 of file types.h.

#### 7.84.2.7 ipv6uc\_transmitted\_octets

`u64 vtss_l3_counters_t::ipv6uc_transmitted_octets`

IPv6UC octets transmitted

Definition at line 894 of file types.h.

#### 7.84.2.8 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`

### 7.85 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`

### 7.85.1 Detailed Description

Neighbour entry.

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

### 7.85.2 Field Documentation

#### 7.85.2.1 dmac

`vtss_mac_t vtss_l3_neighbour_t::dmac`

MAC address of destination

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

#### 7.85.2.2 vlan

`vtss_vid_t vtss_l3_neighbour_t::vlan`

VLAN of destination

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

#### 7.85.2.3 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`

## 7.86 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`

### 7.86.1 Detailed Description

Router leg control structure.

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

### 7.86.2 Field Documentation

#### 7.86.2.1 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`.

#### 7.86.2.2 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`.

#### 7.86.2.3 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.

#### 7.86.2.4 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.

#### 7.86.2.5 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.

#### 7.86.2.6 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.

### 7.86.2.7 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.

### 7.86.2.8 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.

### 7.86.2.9 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](#)

## 7.87 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](#)

### 7.87.1 Detailed Description

Structure for Get PHY LC-PLL status.

Definition at line 1777 of file vtss\_phy\_api.h.

### 7.87.2 Field Documentation

#### 7.87.2.1 lock\_status

```
u8 vtss_lcppll_status_t::lock_status
```

PLL lock status

Definition at line 1778 of file vtss\_phy\_api.h.

#### 7.87.2.2 cal\_done

```
u8 vtss_lcppll_status_t::cal_done
```

Calibration status

Definition at line 1779 of file vtss\_phy\_api.h.

#### 7.87.2.3 cal\_error

```
u8 vtss_lcppll_status_t::cal_error
```

Calibration Error indication

Definition at line 1780 of file vtss\_phy\_api.h.

#### 7.87.2.4 fsm\_lock

```
u8 vtss_lcppll_status_t::fsm_lock
```

FSM lock status

Definition at line 1781 of file vtss\_phy\_api.h.

### 7.87.2.5 fsm\_stat

```
u8 vtss_lcp11_status_t::fsm_stat
```

FSM internal status

Definition at line 1782 of file vtss\_phy\_api.h.

### 7.87.2.6 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](#)

## 7.88 vtss\_learn\_mode\_t Struct Reference

Learning mode.

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

### Data Fields

- [BOOL automatic](#)
- [BOOL cpu](#)
- [BOOL discard](#)

### 7.88.1 Detailed Description

Learning mode.

Definition at line 379 of file vtss\_l2\_api.h.

### 7.88.2 Field Documentation

### 7.88.2.1 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.

### 7.88.2.2 cpu

`BOOL vtss_learn_mode_t::cpu`

Learn frames copied to CPU (default disabled)

Definition at line 382 of file vtss\_l2\_api.h.

### 7.88.2.3 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`

## 7.89 vtss\_mac\_t Struct Reference

MAC Address.

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

### Data Fields

- `u8 addr [6]`

### 7.89.1 Detailed Description

MAC Address.

Definition at line 643 of file types.h.

## 7.89.2 Field Documentation

### 7.89.2.1 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](#)

## 7.90 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](#)

### 7.90.1 Detailed Description

MAC address entry.

Definition at line 119 of file vtss\_l2\_api.h.

## 7.90.2 Field Documentation

### 7.90.2.1 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.

### 7.90.2.2 destination

`BOOL` `vtss_mac_table_entry_t::destination[VTSS_PORT_ARRAY_SIZE]`

Dest. ports

Definition at line 122 of file vtss\_l2\_api.h.

### 7.90.2.3 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.

### 7.90.2.4 locked

`BOOL` `vtss_mac_table_entry_t::locked`

Locked/static flag

Definition at line 124 of file vtss\_l2\_api.h.

### 7.90.2.5 aged

`BOOL` `vtss_mac_table_entry_t::aged`

Age flag

Definition at line 125 of file vtss\_l2\_api.h.

### 7.90.2.6 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.

### 7.90.2.7 enable

`BOOL` `vtss_mac_table_entry_t::enable`

Use (UPSID, UPSPN) when adding entry

Definition at line 132 of file vtss\_l2\_api.h.

### 7.90.2.8 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.

### 7.90.2.9 upsid

`vtss_vstax_upsid_t` `vtss_mac_table_entry_t::upsid`

UPS identifier

Definition at line 134 of file vtss\_l2\_api.h.

### 7.90.2.10 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.

### 7.90.2.11 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](#)

## 7.91 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](#)

### 7.91.1 Detailed Description

MAC address table status.

Definition at line 358 of file [vtss\\_l2\\_api.h](#).

### 7.91.2 Field Documentation

#### 7.91.2.1 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](#).

### 7.91.2.2 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`.

### 7.91.2.3 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`.

### 7.91.2.4 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`

## 7.92 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`

### 7.92.1 Detailed Description

Mirror configuration.

Definition at line 1681 of file vtss\_l2\_api.h.

### 7.92.2 Field Documentation

#### 7.92.2.1 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.

#### 7.92.2.2 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.

#### 7.92.2.3 tag

`vtss_mirror_tag_t` `vtss_mirror_conf_t::tag`

Mirror tag type

Definition at line 1686 of file vtss\_l2\_api.h.

#### 7.92.2.4 vid

`vtss_vid_t` `vtss_mirror_conf_t::vid`

Mirror tag VID

Definition at line 1687 of file vtss\_l2\_api.h.

### 7.92.2.5 pcp

`vtss_tagprior_t vtss_mirror_conf_t::pcp`

Mirror tag PCP

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

### 7.92.2.6 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`

## 7.93 `vtss_mtimer_t` Struct Reference

Timer structure.

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

### Data Fields

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

### 7.93.1 Detailed Description

Timer structure.

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

### 7.93.2 Field Documentation

### 7.93.2.1 timeout

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

Timeout

Definition at line 89 of file vtss\_os\_linux.h.

### 7.93.2.2 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](#)

## 7.94 vtss\_npi\_conf\_t Struct Reference

NPI configuration.

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

### Data Fields

- [BOOL enable](#)
- [vtss\\_port\\_no\\_t port\\_no](#)

### 7.94.1 Detailed Description

NPI configuration.

Definition at line 48 of file vtss\_packet\_api.h.

### 7.94.2 Field Documentation

#### 7.94.2.1 enable

`BOOL vtss_npi_conf_t::enable`

Enable NPI port

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

#### 7.94.2.2 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`

## 7.95 vtss\_os\_timestamp\_t Struct Reference

`#include <vtss_misc_api.h>`

### Data Fields

- `unsigned int hw_cnt`

#### 7.95.1 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`.

#### 7.95.2 Field Documentation

### 7.95.2.1 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](#)

## 7.96 vtss\_packet\_dma\_conf\_t Struct Reference

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

### Data Fields

- [BOOL dma\\_enable \[VTSS\\_QUEUE\\_ARRAY\\_SIZE\]](#)

### 7.96.1 Detailed Description

The packet DMA configuration.

Definition at line 2124 of file vtss\_packet\_api.h.

### 7.96.2 Field Documentation

#### 7.96.2.1 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](#)

## 7.97 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`

#### 7.97.1 Detailed Description

Information about frame.

Definition at line 346 of file vtss\_packet\_api.h.

#### 7.97.2 Field Documentation

##### 7.97.2.1 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.

##### 7.97.2.2 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.

**7.97.2.3 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.

**7.97.2.4 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.

**7.97.2.5 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.

**7.97.2.6 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](#)

## 7.98 **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`

### 7.98.1 Detailed Description

Packet information for each port.

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

### 7.98.2 Field Documentation

#### 7.98.2.1 filter

`vtss_packet_filter_t vtss_packet_port_filter_t::filter`

Packet filtering

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

#### 7.98.2.2 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`

## 7.99 `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`

### 7.99.1 Detailed Description

Port info structure.

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

### 7.99.2 Field Documentation

#### 7.99.2.1 `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`.

#### 7.99.2.2 `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`.

#### 7.99.2.3 `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`.

#### 7.99.2.4 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.

#### 7.99.2.5 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](#)

## 7.100 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]

#### 7.100.1 Detailed Description

CPU Rx configuration.

Definition at line 121 of file vtss\_packet\_api.h.

#### 7.100.2 Field Documentation

### 7.100.2.1 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.

### 7.100.2.2 reg

```
vtss_packet_rx_reg_t vtss_packet_rx_conf_t::reg
```

Packet registration

Definition at line 123 of file vtss\_packet\_api.h.

### 7.100.2.3 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.

### 7.100.2.4 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](#)

## 7.101 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`

### 7.101.1 Detailed Description

System frame header describing received frame.

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

### 7.101.2 Field Documentation

#### 7.101.2.1 `length`

`u32 vtss_packet_rx_header_t::length`

Frame length excluding CRC

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

#### 7.101.2.2 `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`.

#### 7.101.2.3 `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`.

### 7.101.2.4 learn

`BOOL vtss_packet_rx_header_t::learn`

TRUE if learn frame

Definition at line 221 of file `vtss_packet_api.h`.

### 7.101.2.5 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`.

### 7.101.2.6 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`.

### 7.101.2.7 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`

## 7.102 **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`

### 7.102.1 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`.

### 7.102.2 Field Documentation

### 7.102.2.1 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`.

### 7.102.2.2 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`.

### 7.102.2.3 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`.

#### 7.102.2.4 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.

#### 7.102.2.5 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.

### 7.102.2.6 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.

### 7.102.2.7 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.

### 7.102.2.8 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.

### 7.102.2.9 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.

### 7.102.2.10 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.

## 7.102.2.11 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.

## 7.102.2.12 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.

## 7.102.2.13 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.

#### 7.102.2.14 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`.

#### 7.102.2.15 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`.

#### 7.102.2.16 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`.

### 7.102.2.17 sflow\_type

```
vtss_sfloop_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.

### 7.102.2.18 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.

### 7.102.2.19 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 lsb bits 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.

### 7.102.2.20 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.

### 7.102.2.21 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.

### 7.102.2.22 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

## 7.103 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`

### 7.103.1 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.

### 7.103.2 Field Documentation

#### 7.103.2.1 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.

#### 7.103.2.2 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.

### 7.103.2.3 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.

### 7.103.2.4 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.

### 7.103.2.5 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`.

### 7.103.2.6 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`.

### 7.103.2.7 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`

## 7.104 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]

### 7.104.1 Detailed Description

Packet registration per port.

Definition at line 764 of file types.h.

## 7.104.2 Field Documentation

### 7.104.2.1 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.

### 7.104.2.2 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`

## 7.105 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`

### 7.105.1 Detailed Description

CPU Rx queue configuration.

Definition at line 85 of file vtss\_packet\_api.h.

### 7.105.2 Field Documentation

### 7.105.2.1 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.

### 7.105.2.2 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`

## 7.106 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`

### 7.106.1 Detailed Description

CPU Rx queue map.

Definition at line 103 of file vtss\_packet\_api.h.

## 7.106.2 Field Documentation

### 7.106.2.1 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`.

### 7.106.2.2 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`.

### 7.106.2.3 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`.

### 7.106.2.4 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`.

### 7.106.2.5 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`.

### 7.106.2.6 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.

### 7.106.2.7 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.

### 7.106.2.8 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.

### 7.106.2.9 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.

### 7.106.2.10 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.

### 7.106.2.11 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`

## 7.107 `vtss_packet_rx_queue_npi_conf_t` Struct Reference

CPU Rx queue NPI configuration.

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

### Data Fields

- `BOOL enable`

### 7.107.1 Detailed Description

CPU Rx queue NPI configuration.

Definition at line 75 of file `vtss_packet_api.h`.

### 7.107.2 Field Documentation

#### 7.107.2.1 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`

## 7.108 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`

#### 7.108.1 Detailed Description

CPU Rx packet registration.

Definition at line 93 of file vtss\_packet\_api.h.

#### 7.108.2 Field Documentation

##### 7.108.2.1 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.

##### 7.108.2.2 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.

### 7.108.2.3 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.

### 7.108.2.4 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.

### 7.108.2.5 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`

## 7.109 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]`

### 7.109.1 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.

## 7.109.2 Field Documentation

### 7.109.2.1 length

`u32 vtss_packet_tx_ifh_t::length`

Length of compiled IFH (in bytes)

Definition at line 2073 of file vtss\_packet\_api.h.

### 7.109.2.2 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`

## 7.110 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`
} vstax
- `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`

### 7.110.1 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.

### 7.110.2 Field Documentation

#### 7.110.2.1 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.

### 7.110.2.2 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.

### 7.110.2.3 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.

#### 7.110.2.4 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`.

#### 7.110.2.5 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.

#### 7.110.2.6 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.

### 7.110.2.7 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.

### 7.110.2.8 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.

### 7.110.2.9 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.

### 7.110.2.10 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.

### 7.110.2.11 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.

### 7.110.2.12 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.

### 7.110.2.13 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.

### 7.110.2.14 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.

### 7.110.2.15 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.

### 7.110.2.16 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`.

### 7.110.2.17 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`.

## 7.110.2.18 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.

## 7.110.2.19 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.

### 7.110.2.20 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`

## 7.111 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`

### 7.111.1 Detailed Description

10G Phy APC configuration

Definition at line 241 of file `vtss_phy_10g_api.h`.

### 7.111.2 Field Documentation

### 7.111.2.1 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`.

### 7.111.2.2 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`

## 7.112 **vtss\_phy\_10g\_apc\_status\_t** Struct Reference

10G Phy APC status

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

### Data Fields

- `BOOL reset`
- `BOOL freeze`

### 7.112.1 Detailed Description

10G Phy APC status

Definition at line 247 of file `vtss_phy_10g_api.h`.

### 7.112.2 Field Documentation

### 7.112.2.1 reset

`BOOL vtss_phy_10g_apc_status_t::reset`

APC reset status

Definition at line 248 of file `vtss_phy_10g_api.h`.

### 7.112.2.2 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`

## 7.113 `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`

### 7.113.1 Detailed Description

10G PHY Automatic Failover configuration

Definition at line 1807 of file `vtss_phy_10g_api.h`.

## 7.113.2 Field Documentation

### 7.113.2.1 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.

### 7.113.2.2 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.

### 7.113.2.3 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.

### 7.113.2.4 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.

### 7.113.2.5 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.

### 7.113.2.6 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`.

### 7.113.2.7 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`.

### 7.113.2.8 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`.

### 7.113.2.9 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`.

### 7.113.2.10 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`

## 7.114 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`

#### 7.114.1 Detailed Description

10G Phy Base KR Autoneg config

Definition at line 1207 of file vtss\_phy\_10g\_api.h.

#### 7.114.2 Field Documentation

##### 7.114.2.1 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.

##### 7.114.2.2 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.

### 7.114.2.3 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`

## 7.115 `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`

### 7.115.1 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`.

### 7.115.2 Field Documentation

#### 7.115.2.1 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`.

### 7.115.2.2 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.

### 7.115.2.3 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.

### 7.115.2.4 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.

### 7.115.2.5 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.

### 7.115.2.6 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.

### 7.115.2.7 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`

## 7.116 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`

### 7.116.1 Detailed Description

10G Phy Base Link Advertisement capability config

Definition at line 1223 of file `vtss_phy_10g_api.h`.

### 7.116.2 Field Documentation

#### 7.116.2.1 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`.

### 7.116.2.2 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.

### 7.116.2.3 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.

### 7.116.2.4 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](#)

## 7.117 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

### 7.117.1 Detailed Description

10G Phy Base KR Training & Autoneg status

Definition at line 1268 of file vtss\_phy\_10g\_api.h.

## 7.117.2 Field Documentation

### 7.117.2.1 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`.

### 7.117.2.2 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`.

### 7.117.2.3 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`

## 7.118 `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`

### 7.118.1 Detailed Description

10G Phy Base KR Training & Autoneg config

Definition at line 1231 of file vtss\_phy\_10g\_api.h.

### 7.118.2 Field Documentation

#### 7.118.2.1 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.

#### 7.118.2.2 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.

#### 7.118.2.3 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.

#### 7.118.2.4 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.

### 7.118.2.5 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`

## 7.119 `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`

### 7.119.1 Detailed Description

10G Phy Base KR Training config

Definition at line 1214 of file `vtss_phy_10g_api.h`.

### 7.119.2 Field Documentation

#### 7.119.2.1 enable

`BOOL vtss_phy_10g_base_kr_training_t::enable`

Enable KR training

Definition at line 1215 of file `vtss_phy_10g_api.h`.

**7.119.2.2 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.

**7.119.2.3 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.

**7.119.2.4 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.

**7.119.2.5 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](#)

---

**7.120 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`

### 7.120.1 Detailed Description

10G Phy CKOUT config data

Malibu Only

Definition at line 962 of file `vtss_phy_10g_api.h`.

### 7.120.2 Field Documentation

#### 7.120.2.1 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`.

#### 7.120.2.2 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`.

#### 7.120.2.3 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`.

#### 7.120.2.4 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.

#### 7.120.2.5 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.

#### 7.120.2.6 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](#)

## 7.121 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`

### 7.121.1 Detailed Description

Advertisement control data for Clause 37 aneg.

Definition at line 1507 of file vtss\_phy\_10g\_api.h.

### 7.121.2 Field Documentation

#### 7.121.2.1 fdx

`BOOL vtss_phy_10g_clause_37_adv_t::fdx`

(FD)

Definition at line 1509 of file vtss\_phy\_10g\_api.h.

#### 7.121.2.2 hdx

`BOOL vtss_phy_10g_clause_37_adv_t::hdx`

(HD) ,Not supported

Definition at line 1510 of file vtss\_phy\_10g\_api.h.

#### 7.121.2.3 symmetric\_pause

`BOOL vtss_phy_10g_clause_37_adv_t::symmetric_pause`

(PS1)

Definition at line 1511 of file vtss\_phy\_10g\_api.h.

#### 7.121.2.4 asymmetric\_pause

`BOOL vtss_phy_10g_clause_37_adv_t::asymmetric_pause`

(PS2)

Definition at line 1512 of file vtss\_phy\_10g\_api.h.

### 7.121.2.5 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.

### 7.121.2.6 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.

### 7.121.2.7 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`

## 7.122 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`

### 7.122.1 Detailed Description

Clause 37 Auto-negotiation status for line and host.

Definition at line 1529 of file vtss\_phy\_10g\_api.h.

### 7.122.2 Field Documentation

#### 7.122.2.1 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`.

#### 7.122.2.2 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`

## 7.123 `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`

#### 7.123.1 Detailed Description

Clause 37 control struct.

Definition at line 1537 of file `vtss_phy_10g_api.h`.

## 7.123.2 Field Documentation

### 7.123.2.1 enable

`BOOL vtss_phy_10g_clause_37_control_t::enable`

Enable of Autoneg

Definition at line 1539 of file vtss\_phy\_10g\_api.h.

### 7.123.2.2 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.

### 7.123.2.3 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.

### 7.123.2.4 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.

### 7.123.2.5 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.

### 7.123.2.6 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`

## 7.124 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`

### 7.124.1 Detailed Description

Clause 37 Auto-negotiation status.

Definition at line 1519 of file vtss\_phy\_10g\_api.h.

### 7.124.2 Field Documentation

#### 7.124.2.1 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.

#### 7.124.2.2 complete

`BOOL vtss_phy_10g_clause_37_status_t::complete`

Aneg completion status

Definition at line 1523 of file vtss\_phy\_10g\_api.h.

#### 7.124.2.3 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.

#### 7.124.2.4 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`

## 7.125 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`

#### 7.125.1 Detailed Description

10G Phy CLOCK Source Selection

Definition at line 192 of file vtss\_phy\_10g\_api.h.

## 7.125.2 Field Documentation

### 7.125.2.1 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`

## 7.126 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`

### 7.126.1 Detailed Description

10G Phy Sublayer counters

Definition at line 1676 of file `vtss_phy_10g_api.h`.

## 7.126.2 Field Documentation

### 7.126.2.1 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`

## 7.127 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`

#### 7.127.1 Detailed Description

10G OOS workaround options

Definition at line 2090 of file vtss\_phy\_ts\_api.h.

#### 7.127.2 Field Documentation

##### 7.127.2.1 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.

##### 7.127.2.2 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.

### 7.127.2.3 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](#)

## 7.128 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](#)

### 7.128.1 Detailed Description

Firmware status.

Definition at line 2776 of file vtss\_phy\_10g\_api.h.

### 7.128.2 Field Documentation

#### 7.128.2.1 edc\_fw\_rev

```
u16 vtss_phy_10g_fw_status_t::edc_fw_rev
```

FW revision

Definition at line 2777 of file vtss\_phy\_10g\_api.h.

### 7.128.2.2 edc\_fw\_chksum

`BOOL vtss_phy_10g_fw_status_t::edc_fw_chksum`

FW checksum. Fail=0, Pass=1

Definition at line 2778 of file vtss\_phy\_10g\_api.h.

### 7.128.2.3 icpu\_activity

`BOOL vtss_phy_10g_fw_status_t::icpu_activity`

iCPU activity. Not Running=0, Running=1

Definition at line 2779 of file vtss\_phy\_10g\_api.h.

### 7.128.2.4 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 2780 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\_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`

### 7.129.1 Detailed Description

10G Phy Host clock config data

Malibu Only

Definition at line 1052 of file vtss\_phy\_10g\_api.h.

### 7.129.2 Field Documentation

#### 7.129.2.1 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.

#### 7.129.2.2 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.

#### 7.129.2.3 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`

## 7.130 vtss\_phy\_10g\_i2c\_slave\_conf\_t Struct Reference

10G Phy I2C Master Interface for SFP Module Configuration

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

## Data Fields

- `u8 slave_id`
- `u16 prescale`

### 7.130.1 Detailed Description

10G Phy I2C Master Interface for SFP Module Configuration

Definition at line 2905 of file `vtss_phy_10g_api.h`.

### 7.130.2 Field Documentation

#### 7.130.2.1 `slave_id`

`u8 vtss_phy_10g_i2c_slave_conf_t::slave_id`

I2C Slave ID

Definition at line 2906 of file `vtss_phy_10g_api.h`.

#### 7.130.2.2 `prescale`

`u16 vtss_phy_10g_i2c_slave_conf_t::prescale`

SCL frequency = 156.25 MHZ/5\*(Prescale + 1), 0 to 0x4C are invalid settings.

Definition at line 2907 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_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`

### 7.131.1 Detailed Description

10G Phy IB configuration

Definition at line 213 of file vtss\_phy\_10g\_api.h.

### 7.131.2 Field Documentation

#### 7.131.2.1 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.

#### 7.131.2.2 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.

### 7.131.2.3 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`.

### 7.131.2.4 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`.

### 7.131.2.5 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`.

### 7.131.2.6 agc

`ib_par_cfg` `vtss_phy_10g_ib_conf_t::agc`

AGC value

Definition at line 219 of file `vtss_phy_10g_api.h`.

### 7.131.2.7 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`.

### 7.131.2.8 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.

### 7.131.2.9 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.

### 7.131.2.10 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.

### 7.131.2.11 ld

`u8 vtss_phy_10g_ib_conf_t::ld`

level detect

Definition at line 224 of file vtss\_phy\_10g\_api.h.

### 7.131.2.12 prbs

`u8 vtss_phy_10g_ib_conf_t::prbs`

PRBS RX pattern selected

Definition at line 225 of file vtss\_phy\_10g\_api.h.

### 7.131.2.13 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.

### 7.131.2.14 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.

### 7.131.2.15 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.

### 7.131.2.16 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.

### 7.131.2.17 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](#)

## 7.132 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`

#### 7.132.1 Detailed Description

10G Phy IB configuration

Definition at line 234 of file vtss\_phy\_10g\_api.h.

#### 7.132.2 Field Documentation

##### 7.132.2.1 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.

##### 7.132.2.2 sig\_det

```
BOOL vtss_phy_10g_ib_status_t::sig_det
```

Signal detect

Definition at line 236 of file vtss\_phy\_10g\_api.h.

### 7.132.2.3 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`

## 7.133 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]`

### 7.133.1 Detailed Description

VSCOPE fast scan storage.

Definition at line 1898 of file vtss\_phy\_10g\_api.h.

### 7.133.2 Field Documentation

#### 7.133.2.1 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.

### 7.133.2.2 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](#)

## 7.134 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](#)

### 7.134.1 Detailed Description

10G Phy part number and revision

Definition at line 2269 of file vtss\_phy\_10g\_api.h.

### 7.134.2 Field Documentation

#### 7.134.2.1 part\_number

```
u16 vtss_phy_10g_id_t::part_number
```

Part number (Hex)

Definition at line 2271 of file vtss\_phy\_10g\_api.h.

### 7.134.2.2 revision

`u16 vtss_phy_10g_id_t::revision`

Chip revision

Definition at line 2272 of file vtss\_phy\_10g\_api.h.

### 7.134.2.3 channel\_id

`u16 vtss_phy_10g_id_t::channel_id`

Channel id

Definition at line 2273 of file vtss\_phy\_10g\_api.h.

### 7.134.2.4 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.

### 7.134.2.5 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.

### 7.134.2.6 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.

#### 7.134.2.7 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](#)

### 7.135 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](#)

#### 7.135.1 Detailed Description

10G Phy Initialization configuration

Definition at line 432 of file vtss\_phy\_10g\_api.h.

#### 7.135.2 Field Documentation

##### 7.135.2.1 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](#)

## 7.136 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`

#### 7.136.1 Detailed Description

10G Phy Optimisation of jitter performance

Definition at line 302 of file vtss\_phy\_10g\_api.h.

#### 7.136.2 Field Documentation

##### 7.136.2.1 incr\_levn

```
BOOL vtss_phy_10g_jitter_conf_t::incr_levn
```

Increase LevN

Definition at line 303 of file vtss\_phy\_10g\_api.h.

##### 7.136.2.2 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.

### 7.136.2.3 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](#)

## 7.137 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](#)

### 7.137.1 Detailed Description

10G Phy Base KR Autoneg status

Definition at line 1240 of file vtss\_phy\_10g\_api.h.

### 7.137.2 Field Documentation

#### 7.137.2.1 complete

```
BOOL vtss_phy_10g_kr_status_aneg_t::complete
```

Aneg completed successfully

Definition at line 1241 of file vtss\_phy\_10g\_api.h.

### 7.137.2.2 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.

### 7.137.2.3 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.

### 7.137.2.4 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.

### 7.137.2.5 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.

### 7.137.2.6 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.

### 7.137.2.7 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`.

### 7.137.2.8 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`.

### 7.137.2.9 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`

## 7.138 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`

### 7.138.1 Detailed Description

10G Phy Base KR FEC status

Definition at line 1261 of file `vtss_phy_10g_api.h`.

## 7.138.2 Field Documentation

### 7.138.2.1 enable

`BOOL vtss_phy_10g_kr_status_fec_t::enable`

FEC Enabled

Definition at line 1262 of file vtss\_phy\_10g\_api.h.

### 7.138.2.2 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.

### 7.138.2.3 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`

## 7.139 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`

### 7.139.1 Detailed Description

10G Phy Base KR Training status

Definition at line 1253 of file vtss\_phy\_10g\_api.h.

### 7.139.2 Field Documentation

#### 7.139.2.1 complete

`BOOL vtss_phy_10g_kr_status_train_t::complete`

Training completed successfully

Definition at line 1254 of file vtss\_phy\_10g\_api.h.

#### 7.139.2.2 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.

#### 7.139.2.3 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.

#### 7.139.2.4 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](#)

## 7.140 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`

#### 7.140.1 Detailed Description

10G Phy Lane SYNC Configuration

Malibu Only

Definition at line 1130 of file vtss\_phy\_10g\_api.h.

#### 7.140.2 Field Documentation

##### 7.140.2.1 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.

##### 7.140.2.2 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.

### 7.140.2.3 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.

### 7.140.2.4 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.

### 7.140.2.5 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

## 7.141 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](#)

### 7.141.1 Detailed Description

10G Phy Line clock config data

Malibu Only

Definition at line 1026 of file vtss\_phy\_10g\_api.h.

### 7.141.2 Field Documentation

#### 7.141.2.1 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.

#### 7.141.2.2 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.

#### 7.141.2.3 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`

## 7.142 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`

### 7.142.1 Detailed Description

10G Phy system and network loopbacks

Definition at line 1625 of file `vtss_phy_10g_api.h`.

### 7.142.2 Field Documentation

#### 7.142.2.1 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`.

#### 7.142.2.2 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`

## 7.143 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`

### 7.143.1 Detailed Description

10G Phy operating mode

Definition at line 333 of file vtss\_phy\_10g\_api.h.

### 7.143.2 Member Enumeration Documentation

#### 7.143.2.1 anonymous enum

anonymous enum

EDC modes.

Enumerator

VTSS_EDC_FW_LOAD_MDIO	Load EDC FW through MDIO to iCPU
VTSS_EDC_FW_LOAD_NOTHING	Do not load FW to iCPU

Definition at line 357 of file vtss\_phy\_10g\_api.h.

### 7.143.3 Field Documentation

#### 7.143.3.1 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.

#### 7.143.3.2 interface

`vtss_phy_interface_mode vtss_phy_10g_mode_t::interface`

Interface mode.

Definition at line 336 of file vtss\_phy\_10g\_api.h.

### 7.143.3.3 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.

### 7.143.3.4 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.

### 7.143.3.5 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.

### 7.143.3.6 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.

### 7.143.3.7 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.

### 7.143.3.8 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.

### 7.143.3.9 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.

### 7.143.3.10 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.

### 7.143.3.11 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.

### 7.143.3.12 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.

#### 7.143.3.13 edc\_fw\_load

```
enum { ... } vtss_phy_10g_mode_t::edc_fw_load
```

EDC modes.

EDC Firmware load

#### 7.143.3.14 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.

#### 7.143.3.15 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.

#### 7.143.3.16 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.

#### 7.143.3.17 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.

### 7.143.3.18 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.

### 7.143.3.19 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.

### 7.143.3.20 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.

### 7.143.3.21 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.

### 7.143.3.22 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.

### 7.143.3.23 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.

### 7.143.3.24 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.

### 7.143.3.25 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.

### 7.143.3.26 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.

### 7.143.3.27 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.

### 7.143.3.28 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.

### 7.143.3.29 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.

### 7.143.3.30 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.

### 7.143.3.31 serdes\_conf

`struct { ... } vtss_phy_10g_mode_t::serdes_conf`

Serdes parameters.

Serdes configuration

### 7.143.3.32 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.

### 7.143.3.33 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.

### 7.143.3.34 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.

### 7.143.3.35 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.

### 7.143.3.36 master

`clk_mstr_t vtss_phy_10g_mode_t::master`

Clock Master

Definition at line 390 of file vtss\_phy\_10g\_api.h.

### 7.143.3.37 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.

### 7.143.3.38 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.

### 7.143.3.39 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.

### 7.143.3.40 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.

### 7.143.3.41 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.

### 7.143.3.42 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.

#### 7.143.3.43 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.

#### 7.143.3.44 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.

#### 7.143.3.45 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.

#### 7.143.3.46 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.

#### 7.143.3.47 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.

#### 7.143.3.48 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`.

#### 7.143.3.49 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`.

#### 7.143.3.50 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`.

#### 7.143.3.51 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`.

#### 7.143.3.52 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`

## 7.144 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`

### 7.144.1 Detailed Description

10G Phy OB status

Definition at line 1171 of file vtss\_phy\_10g\_api.h.

### 7.144.2 Field Documentation

#### 7.144.2.1 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.

#### 7.144.2.2 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.

#### 7.144.2.3 slew

`u8 vtss_phy_10g_ob_status_t::slew`

slew rate

Definition at line 1174 of file `vtss_phy_10g_api.h`.

#### 7.144.2.4 levn

`u8 vtss_phy_10g_ob_status_t::levn`

amplitude

Definition at line 1175 of file `vtss_phy_10g_api.h`.

#### 7.144.2.5 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`.

#### 7.144.2.6 v3

`int vtss_phy_10g_ob_status_t::v3`

d\_filter tap v3

Definition at line 1177 of file `vtss_phy_10g_api.h`.

#### 7.144.2.7 vp

`int vtss_phy_10g_ob_status_t::vp`

d\_filter tap vp

Definition at line 1178 of file `vtss_phy_10g_api.h`.

## 7.144.2.8 v4

```
int vtss_phy_10g_ob_status_t::v4
```

d\_filter tap v4

Definition at line 1179 of file vtss\_phy\_10g\_api.h.

## 7.144.2.9 v5

```
int vtss_phy_10g_ob_status_t::v5
```

d\_filter tap v5

Definition at line 1180 of file vtss\_phy\_10g\_api.h.

## 7.144.2.10 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](#)

## 7.145 vtss\_phy\_10g\_pcs\_prbs\_gen\_conf\_t Struct Reference

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

### Data Fields

- [BOOL prbs\\_gen](#)

#### 7.145.1 Detailed Description

\ brief 10G PHY pcs prbs generator configuration

Definition at line 1941 of file vtss\_phy\_10g\_api.h.

---

## 7.145.2 Field Documentation

### 7.145.2.1 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`

## 7.146 `vtss_phy_10g_pcs_prbs_mon_conf_t` Struct Reference

`#include <vtss_phy_10g_api.h>`

### Data Fields

- `BOOL prbs_mon`
- `u32 error_counter`

### 7.146.1 Detailed Description

\ brief 10G PHY pcs prbs analyzer configuration

Definition at line 1976 of file `vtss_phy_10g_api.h`.

### 7.146.2 Field Documentation

#### 7.146.2.1 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`.

### 7.146.2.2 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](#)

## 7.147 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](#)

### 7.147.1 Detailed Description

10G PHY Packet generator configuration

Definition at line 2133 of file vtss\_phy\_10g\_api.h.

### 7.147.2 Field Documentation

#### 7.147.2.1 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`.

#### 7.147.2.2 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`.

#### 7.147.2.3 ingress

`BOOL vtss_phy_10g_pkt_gen_conf_t::ingress`

Ingress or egress

Definition at line 2136 of file `vtss_phy_10g_api.h`.

#### 7.147.2.4 frames

`BOOL vtss_phy_10g_pkt_gen_conf_t::frames`

frames or idles

Definition at line 2137 of file `vtss_phy_10g_api.h`.

#### 7.147.2.5 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`.

### 7.147.2.6 etype

`u16 vtss_phy_10g_pkt_gen_conf_t::etype`

Ethertype

Definition at line 2139 of file vtss\_phy\_10g\_api.h.

### 7.147.2.7 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.

### 7.147.2.8 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.

### 7.147.2.9 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.

### 7.147.2.10 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.

#### 7.147.2.11 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.

#### 7.147.2.12 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.

#### 7.147.2.13 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

## 7.148 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`

### 7.148.1 Detailed Description

10G PHY Packet Monitor configuration

Definition at line 2178 of file vtss\_phy\_10g\_api.h.

### 7.148.2 Field Documentation

#### 7.148.2.1 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.

#### 7.148.2.2 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.

#### 7.148.2.3 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.

#### 7.148.2.4 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.

#### 7.148.2.5 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`.

#### 7.148.2.6 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`.

#### 7.148.2.7 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`.

#### 7.148.2.8 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`

### 7.149 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`

### 7.149.1 Detailed Description

10G Phy Polarity inversion

Definition at line 162 of file `vtss_phy_10g_api.h`.

### 7.149.2 Field Documentation

#### 7.149.2.1 `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`.

#### 7.149.2.2 `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`.

#### 7.149.2.3 `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`.

#### 7.149.2.4 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`

## 7.150 `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`

#### 7.150.1 Detailed Description

\ brief 10G PHY prbs generator configuration

Definition at line 2026 of file `vtss_phy_10g_api.h`.

#### 7.150.2 Field Documentation

##### 7.150.2.1 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`.

### 7.150.2.2 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.

### 7.150.2.3 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.

### 7.150.2.4 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.

### 7.150.2.5 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](#)

## 7.151 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`

### 7.151.1 Detailed Description

10G PHY prbs monitor Configuration

Definition at line 2065 of file vtss\_phy\_10g\_api.h.

### 7.151.2 Field Documentation

#### 7.151.2.1 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.

#### 7.151.2.2 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.

### 7.151.2.3 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.

### 7.151.2.4 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.

### 7.151.2.5 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.

### 7.151.2.6 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.

### 7.151.2.7 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.

### 7.151.2.8 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.

### 7.151.2.9 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.

### 7.151.2.10 error\_status

`u32 vtss_phy_10g_prbs_mon_conf_t::error_status`

Error stautos of PRBS

Definition at line 2075 of file vtss\_phy\_10g\_api.h.

### 7.151.2.11 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.

### 7.151.2.12 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.

### 7.151.2.13 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.

### 7.151.2.14 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.

### 7.151.2.15 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.

### 7.151.2.16 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.

### 7.151.2.17 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.

### 7.151.2.18 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`

## 7.152 `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`

### 7.152.1 Detailed Description

10G Phy RXCKOUT config data

Definition at line 706 of file `vtss_phy_10g_api.h`.

### 7.152.2 Field Documentation

#### 7.152.2.1 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`.

### 7.152.2.2 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.

### 7.152.2.3 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

## 7.153 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`

### 7.153.1 Detailed Description

10G Phy SCKOUT config data

Malibu Only

Definition at line 982 of file vtss\_phy\_10g\_api.h.

### 7.153.2 Field Documentation

### 7.153.2.1 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`.

### 7.153.2.2 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`.

### 7.153.2.3 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`.

### 7.153.2.4 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`.

### 7.153.2.5 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`

## 7.154 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

### 7.154.1 Detailed Description

10G Phy SERDES status

Definition at line 253 of file vtss\_phy\_10g\_api.h.

### 7.154.2 Field Documentation

### 7.154.2.1 rcomp

`u8 vtss_phy_10g_serdes_status_t::rcomp`

Measured Resistor value

Definition at line 254 of file vtss\_phy\_10g\_api.h.

### 7.154.2.2 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.

### 7.154.2.3 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.

### 7.154.2.4 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.

### 7.154.2.5 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.

#### 7.154.2.6 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.

#### 7.154.2.7 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.

#### 7.154.2.8 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.

#### 7.154.2.9 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.

#### 7.154.2.10 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.

#### 7.154.2.11 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.

#### 7.154.2.12 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.

#### 7.154.2.13 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.

#### 7.154.2.14 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.

#### 7.154.2.15 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.

**7.154.2.16 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.

**7.154.2.17 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.

**7.154.2.18 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.

**7.154.2.19 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.

**7.154.2.20 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.

#### 7.154.2.21 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.

#### 7.154.2.22 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.

#### 7.154.2.23 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.

#### 7.154.2.24 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.

#### 7.154.2.25 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.

#### 7.154.2.26 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`.

#### 7.154.2.27 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`.

#### 7.154.2.28 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`.

#### 7.154.2.29 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`.

#### 7.154.2.30 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`

## 7.155 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`

#### 7.155.1 Detailed Description

10G Phy srefclk config data

Definition at line 787 of file vtss\_phy\_10g\_api.h.

#### 7.155.2 Field Documentation

##### 7.155.2.1 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.

##### 7.155.2.2 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](#)

## 7.156 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`

### 7.156.1 Detailed Description

10G Phy link and fault status for all sublayers

Definition at line 1428 of file `vtss_phy_10g_api.h`.

### 7.156.2 Field Documentation

#### 7.156.2.1 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`.

#### 7.156.2.2 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`.

#### 7.156.2.3 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`.

#### 7.156.2.4 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.

#### 7.156.2.5 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.

#### 7.156.2.6 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.

#### 7.156.2.7 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.

#### 7.156.2.8 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.

### 7.156.2.9 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.

### 7.156.2.10 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.

### 7.156.2.11 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](#)

## 7.157 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]`

### 7.157.1 Detailed Description

10G PHY timestamp value array(holder)

Definition at line 2190 of file vtss\_phy\_10g\_api.h.

## 7.157.2 Field Documentation

### 7.157.2.1 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](#)

## 7.158 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](#)

### 7.158.1 Detailed Description

10G Phy TXCKOUT config data

Definition at line 743 of file vtss\_phy\_10g\_api.h.

## 7.158.2 Field Documentation

### 7.158.2.1 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](#)

## 7.159 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`

#### 7.159.1 Detailed Description

\ brief VSCOPE scan configuration

Definition at line 1856 of file vtss\_phy\_10g\_api.h.

#### 7.159.2 Field Documentation

##### 7.159.2.1 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.

##### 7.159.2.2 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.

##### 7.159.2.3 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.

#### 7.159.2.4 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`

## 7.160 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`

#### 7.160.1 Detailed Description

VSCOPE scan configuration.

Definition at line 1904 of file vtss\_phy\_10g\_api.h.

#### 7.160.2 Field Documentation

##### 7.160.2.1 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.

### 7.160.2.2 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.

### 7.160.2.3 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.

### 7.160.2.4 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.

### 7.160.2.5 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.

### 7.160.2.6 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.

### 7.160.2.7 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`.

### 7.160.2.8 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`

## 7.161 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]`

### 7.161.1 Detailed Description

\ brief Vsphere eye scan status

Definition at line 1919 of file `vtss_phy_10g_api.h`.

### 7.161.2 Field Documentation

### 7.161.2.1 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.

### 7.161.2.2 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.

### 7.161.2.3 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.

### 7.161.2.4 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.

### 7.161.2.5 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](#)

## 7.162 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`

### 7.162.1 Detailed Description

PHY auto negotiation advertisement.

Definition at line 309 of file `vtss_phy_api.h`.

### 7.162.2 Field Documentation

#### 7.162.2.1 speed\_10m\_hdx

```
BOOL vtss_phy_aneg_t::speed_10m_hdx
```

10Mbps, half duplex

Definition at line 310 of file `vtss_phy_api.h`.

#### 7.162.2.2 speed\_10m\_fdx

```
BOOL vtss_phy_aneg_t::speed_10m_fdx
```

10Mbps, full duplex

Definition at line 311 of file `vtss_phy_api.h`.

### 7.162.2.3 speed\_100m\_hdx

`BOOL vtss_phy_aneg_t::speed_100m_hdx`

100Mbps, half duplex

Definition at line 312 of file vtss\_phy\_api.h.

### 7.162.2.4 speed\_100m\_fdx

`BOOL vtss_phy_aneg_t::speed_100m_fdx`

100Mbps, full duplex

Definition at line 313 of file vtss\_phy\_api.h.

### 7.162.2.5 speed\_1g\_fdx

`BOOL vtss_phy_aneg_t::speed_1g_fdx`

1000Mbps, full duplex

Definition at line 314 of file vtss\_phy\_api.h.

### 7.162.2.6 speed\_1g\_hdx

`BOOL vtss_phy_aneg_t::speed_1g_hdx`

1000Mbps, full duplex

Definition at line 315 of file vtss\_phy\_api.h.

### 7.162.2.7 symmetric\_pause

`BOOL vtss_phy_aneg_t::symmetric_pause`

Symmetric pause

Definition at line 316 of file vtss\_phy\_api.h.

### 7.162.2.8 asymmetric\_pause

`BOOL vtss_phy_aneg_t::asymmetric_pause`

Asymmetric pause

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

### 7.162.2.9 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`

## 7.163 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`

### 7.163.1 Detailed Description

PHY clock configuration.

Definition at line 648 of file `vtss_phy_api.h`.

### 7.163.2 Field Documentation

### 7.163.2.1 src

`vtss_phy_clk_source_t` `vtss_phy_clock_conf_t::src`

Clock source

Definition at line 649 of file `vtss_phy_api.h`.

### 7.163.2.2 freq

`vtss_phy_freq_t` `vtss_phy_clock_conf_t::freq`

Clock frequency

Definition at line 650 of file `vtss_phy_api.h`.

### 7.163.2.3 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`

## 7.164 vtss\_phy\_conf\_1g\_t Struct Reference

PHY 1G configuration.

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

### Data Fields

- struct {  
    BOOL cfg  
    BOOL val  
} master

### 7.164.1 Detailed Description

PHY 1G configuration.

Definition at line 504 of file vtss\_phy\_api.h.

### 7.164.2 Field Documentation

#### 7.164.2.1 cfg

`BOOL vtss_phy_conf_1g_t::cfg`

Manual Master/Slave Config. 1=enabled

Definition at line 506 of file vtss\_phy\_api.h.

#### 7.164.2.2 val

`BOOL vtss_phy_conf_1g_t::val`

Master/Slave Config value, 1=Master

Definition at line 507 of file vtss\_phy\_api.h.

#### 7.164.2.3 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](#)

## 7.165 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`
- `BOOL skip_coma`

### 7.165.1 Detailed Description

PHY configuration.

Definition at line 395 of file vtss\_phy\_api.h.

### 7.165.2 Field Documentation

#### 7.165.2.1 mode

`vtss_phy_mode_t vtss_phy_conf_t::mode`

PHY mode

Definition at line 396 of file vtss\_phy\_api.h.

#### 7.165.2.2 forced

`vtss_phy_forced_t vtss_phy_conf_t::forced`

Forced mode configuration

Definition at line 397 of file vtss\_phy\_api.h.

#### 7.165.2.3 aneg

`vtss_phy_aneg_t vtss_phy_conf_t::aneg`

Auto-negotiation mode configuration

Definition at line 398 of file vtss\_phy\_api.h.

#### 7.165.2.4 mdi

`vtss_phy_mdi_t` `vtss_phy_conf_t::mdi`

Cu cable MDI (Crossed cable / normal cable)

Definition at line 399 of file `vtss_phy_api.h`.

#### 7.165.2.5 flf

`vtss_phy_fast_link_fail_t` `vtss_phy_conf_t::flf`

Fast link failure configuration

Definition at line 400 of file `vtss_phy_api.h`.

#### 7.165.2.6 sigdet

`vtss_phy_sigdet_polarity_t` `vtss_phy_conf_t::sigdet`

Sigdet pin polarity configuration

Definition at line 401 of file `vtss_phy_api.h`.

#### 7.165.2.7 unidir

`vtss_phy_unidirectional_t` `vtss_phy_conf_t::unidir`

Unidirectional Configuration

Definition at line 402 of file `vtss_phy_api.h`.

#### 7.165.2.8 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 403 of file `vtss_phy_api.h`.

### 7.165.2.9 media\_if\_pcs

`vtss_phy_media_serdes_ctrl_t vtss_phy_conf_t::media_if_pcs`

PHY MAC SerDes PCS Control (Reg23E3)

Definition at line 404 of file `vtss_phy_api.h`.

### 7.165.2.10 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 405 of file `vtss_phy_api.h`.

### 7.165.2.11 skip\_coma

`BOOL vtss_phy_conf_t::skip_coma`

PHY COMA Mode - Automatically apply COMA Config or SKIP

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`

## 7.166 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`

### 7.166.1 Detailed Description

SPI daisy chain configuration.

Definition at line 535 of file `vtss_phy_ts_api.h`.

## 7.166.2 Field Documentation

### 7.166.2.1 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`.

### 7.166.2.2 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`

## 7.167 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`

### 7.167.1 Detailed Description

EEE configuration.

Definition at line 987 of file `vtss_phy_api.h`.

### 7.167.2 Field Documentation

### 7.167.2.1 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.

### 7.167.2.2 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`

## 7.168 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`

### 7.168.1 Detailed Description

enhanced LED control

Definition at line 1010 of file vtss\_phy\_api.h.

### 7.168.2 Field Documentation

#### 7.168.2.1 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`.

#### 7.168.2.2 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`.

#### 7.168.2.3 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`.

#### 7.168.2.4 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, 0x2 = 2 LEDs, 0x3 = 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`

## 7.169 vtss\_phy\_forced\_t Struct Reference

PHY forced mode configuration.

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

## Data Fields

- `vtss_port_speed_t speed`
- `BOOL fdx`

### 7.169.1 Detailed Description

PHY forced mode configuration.

Definition at line 303 of file `vtss_phy_api.h`.

### 7.169.2 Field Documentation

#### 7.169.2.1 speed

`vtss_port_speed_t vtss_phy_forced_t::speed`

Speed

Definition at line 304 of file `vtss_phy_api.h`.

#### 7.169.2.2 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`

## 7.170 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`

### 7.170.1 Detailed Description

LED model selection.

Definition at line 136 of file vtss\_phy\_api.h.

### 7.170.2 Field Documentation

#### 7.170.2.1 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.

#### 7.170.2.2 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`

## 7.171 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`

### 7.171.1 Detailed Description

1G Phy loopbacks

Definition at line 1384 of file vtss\_phy\_api.h.

### 7.171.2 Field Documentation

#### 7.171.2.1 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.

#### 7.171.2.2 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.

#### 7.171.2.3 connector\_enable

```
BOOL vtss_phy_loopback_t::connector_enable
```

Enable/Disable loopback connector loopback

Definition at line 1387 of file vtss\_phy\_api.h.

#### 7.171.2.4 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.

#### 7.171.2.5 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`.

#### 7.171.2.6 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`.

#### 7.171.2.7 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`.

#### 7.171.2.8 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`.

#### 7.171.2.9 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`

## 7.172 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`

#### 7.172.1 Detailed Description

Frequency synthesis pulse configuration.

Definition at line 501 of file vtss\_phy\_ts\_api.h.

#### 7.172.2 Field Documentation

##### 7.172.2.1 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.

##### 7.172.2.2 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.

### 7.172.2.3 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

## 7.173 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

### 7.173.1 Detailed Description

PHY MAC SerDes PCS Control, Reg16E3.

Definition at line 351 of file vtss\_phy\_api.h.

### 7.173.2 Field Documentation

### 7.173.2.1 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.

### 7.173.2.2 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.

### 7.173.2.3 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.

### 7.173.2.4 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.

### 7.173.2.5 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.

### 7.173.2.6 sgmii\_in\_pre

```
vtss_phy_mac_serd_pcs_sgmii_pre vtss_phy_mac_serd_pcs_cntl_t::sgmii_in_pre
```

SGMII Input Preamble for 100BaseFX

Definition at line 357 of file vtss\_phy\_api.h.

### 7.173.2.7 sgmii\_out\_pre

```
BOOL vtss_phy_mac_serd_pcs_cntl_t::sgmii_out_pre
```

SGMII Output Preamble

Definition at line 358 of file vtss\_phy\_api.h.

### 7.173.2.8 serdes\_aneg\_ena

```
BOOL vtss_phy_mac_serd_pcs_cntl_t::serdes_aneg_ena
```

MAC SerDes ANEG Enable

Definition at line 359 of file vtss\_phy\_api.h.

### 7.173.2.9 serdes\_pol\_inv\_in

```
BOOL vtss_phy_mac_serd_pcs_cntl_t::serdes_pol_inv_in
```

Invert SerDes Polarity at input of MAC

Definition at line 360 of file vtss\_phy\_api.h.

### 7.173.2.10 serdes\_pol\_inv\_out

```
BOOL vtss_phy_mac_serd_pcs_cntl_t::serdes_pol_inv_out
```

Invert SerDes Polarity at output of MAC

Definition at line 361 of file vtss\_phy\_api.h.

### 7.173.2.11 fast\_link\_stat\_ena

`BOOL vtss_phy_mac_serdes_pcs_ctrl_t::fast_link_stat_ena`

Fast Link Fail Status Enable

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

### 7.173.2.12 inhibit\_odd\_start

`BOOL vtss_phy_mac_serdes_pcs_ctrl_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`

## 7.174 vtss\_phy\_media\_serdes\_pcs\_ctrl\_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`

### 7.174.1 Detailed Description

PHY MEDIA SerDes PCS Control, Reg23E3.

Definition at line 375 of file `vtss_phy_api.h`.

## 7.174.2 Field Documentation

### 7.174.2.1 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.

### 7.174.2.2 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.

### 7.174.2.3 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.

### 7.174.2.4 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.

### 7.174.2.5 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.

#### 7.174.2.6 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`.

#### 7.174.2.7 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`.

#### 7.174.2.8 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`.

#### 7.174.2.9 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`

## 7.175 **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`

### 7.175.1 Detailed Description

10G Phy PCS counters

Definition at line 1668 of file `vtss_phy_10g_api.h`.

### 7.175.2 Field Documentation

#### 7.175.2.1 `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`.

#### 7.175.2.2 `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`.

#### 7.175.2.3 `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`.

### 7.175.2.4 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](#)

## 7.176 vtss\_phy\_power\_conf\_t Struct Reference

PHY power configuration.

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

### Data Fields

- [vtss\\_phy\\_power\\_mode\\_t mode](#)

### 7.176.1 Detailed Description

PHY power configuration.

Definition at line 562 of file vtss\_phy\_api.h.

### 7.176.2 Field Documentation

#### 7.176.2.1 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](#)

## 7.177 vtss\_phy\_power\_status\_t Struct Reference

PHY power status.

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

### Data Fields

- [u32 level](#)

#### 7.177.1 Detailed Description

PHY power status.

Definition at line 597 of file vtss\_phy\_api.h.

#### 7.177.2 Field Documentation

##### 7.177.2.1 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](#)

## 7.178 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](#)

### 7.178.1 Detailed Description

PHY reset structure.

Definition at line 218 of file vtss\_phy\_api.h.

### 7.178.2 Field Documentation

#### 7.178.2.1 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.

#### 7.178.2.2 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.

#### 7.178.2.3 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.

#### 7.178.2.4 tbi

`vtss_phy_tbi_conf_t` vtss\_phy\_reset\_conf\_t::tbi

TBI setup

Definition at line 222 of file vtss\_phy\_api.h.

### 7.178.2.5 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`.

### 7.178.2.6 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`.

### 7.178.2.7 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`

## 7.179 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`

### 7.179.1 Detailed Description

PHY RGMII configuration.

Definition at line 194 of file `vtss_phy_api.h`.

## 7.179.2 Field Documentation

### 7.179.2.1 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.

### 7.179.2.2 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](#)

## 7.180 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

### 7.180.1 Detailed Description

Phy statistic information.

Definition at line 1348 of file vtss\_phy\_api.h.

## 7.180.2 Field Documentation

### 7.180.2.1 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.

### 7.180.2.2 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.

### 7.180.2.3 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.

### 7.180.2.4 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.

#### 7.180.2.5 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.

#### 7.180.2.6 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.

#### 7.180.2.7 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](#)

## 7.181 vtss\_phy\_status\_1g\_t Struct Reference

PHY 1G status.

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

### Data Fields

- `BOOL master_cfg_fault`
- `BOOL master`

### 7.181.1 Detailed Description

PHY 1G status.

Definition at line 538 of file vtss\_phy\_api.h.

### 7.181.2 Field Documentation

#### 7.181.2.1 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.

#### 7.181.2.2 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`

## 7.182 vtss\_phy\_tbi\_conf\_t Struct Reference

PHY TBI configuration.

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

### Data Fields

- `BOOL aneg_enable`

#### 7.182.1 Detailed Description

PHY TBI configuration.

Definition at line 200 of file vtss\_phy\_api.h.

## 7.182.2 Field Documentation

### 7.182.2.1 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`

## 7.183 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`

### 7.183.1 Detailed Description

PHY timestamp in seconds and nanoseconds (10 bytes Timestamp)

Definition at line 250 of file vtss\_phy\_ts\_api.h.

## 7.183.2 Field Documentation

### 7.183.2.1 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.

### 7.183.2.2 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`.

### 7.183.2.3 seconds

`struct { ... } vtss_phy_timestamp_t::seconds`

6 bytes second part of Timestamp

### 7.183.2.4 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`

## 7.184 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`

### 7.184.1 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.

### 7.184.2 Field Documentation

#### 7.184.2.1 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.

#### 7.184.2.2 mask [1/2]

```
u8 vtss_phy_ts_ach_conf_t::mask
```

Mask

Definition at line 1103 of file vtss\_phy\_ts\_api.h.

#### 7.184.2.3 version

```
struct { ... } vtss_phy_ts_ach_conf_t::version
```

version number of the PWACH in value/mask; set mask 0 for don't care

#### 7.184.2.4 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.

#### 7.184.2.5 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.

#### 7.184.2.6 channel\_type

```
struct { ... } vtss_phy_ts_ach_conf_t::channel_type
```

PW Associated Channel Type in value/mask format

#### 7.184.2.7 proto\_id

```
struct { ... } vtss_phy_ts_ach_conf_t::proto_id
```

PID: identifier of payload as defined in RFC 5718, only for PTP

#### 7.184.2.8 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](#)

### 7.185 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](#)

### 7.185.1 Detailed Description

parameter for setting the alternative clock mode.

external clock output configuration.

Definition at line 52 of file vtss\_phy\_ts\_api.h.

### 7.185.2 Field Documentation

#### 7.185.2.1 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.

#### 7.185.2.2 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.

#### 7.185.2.3 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`

## 7.186 vtss\_phy\_ts\_eng\_init\_conf\_t Struct Reference

Defines the basic engine parameters passed to the `engine_init_conf_get()` 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`

### 7.186.1 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`.

### 7.186.2 Field Documentation

#### 7.186.2.1 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`.

#### 7.186.2.2 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`.

#### 7.186.2.3 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`.

## 7.186.2.4 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.

## 7.186.2.5 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](#)

## 7.187 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`

### 7.187.1 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.

## 7.187.2 Field Documentation

### 7.187.2.1 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.

### 7.187.2.2 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.

### 7.187.2.3 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.

### 7.187.2.4 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.

### 7.187.2.5 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.

### 7.187.2.6 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`

## 7.188 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`

### 7.188.1 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`.

### 7.188.2 Field Documentation

#### 7.188.2.1 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`.

### 7.188.2.2 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.

### 7.188.2.3 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.

### 7.188.2.4 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.

### 7.188.2.5 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.

### 7.188.2.6 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

## 7.189 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]

### 7.189.1 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.

## 7.189.2 Field Documentation

### 7.189.2.1 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.

### 7.189.2.2 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.

### 7.189.2.3 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.

### 7.189.2.4 comm\_opt

`struct { ... } vtss_phy_ts_eth_conf_t::comm_opt`

Ethernet common config

#### 7.189.2.5 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.

#### 7.189.2.6 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.

#### 7.189.2.7 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.

#### 7.189.2.8 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.

#### 7.189.2.9 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.

#### 7.189.2.10 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`.

#### 7.189.2.11 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`.

#### 7.189.2.12 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`.

#### 7.189.2.13 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`.

#### 7.189.2.14 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`.

**7.189.2.15 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.

**7.189.2.16 range [1/2]**

`struct { ... } vtss_phy_ts_eth_conf_t::range`

tag in range

**7.189.2.17 val [1/2]**

`u16 vtss_phy_ts_eth_conf_t::val`

Value

Definition at line 993 of file vtss\_phy\_ts\_api.h.

**7.189.2.18 mask [1/2]**

`u16 vtss_phy_ts_eth_conf_t::mask`

Mask

Definition at line 994 of file vtss\_phy\_ts\_api.h.

**7.189.2.19 value [1/2]**

`struct { ... } vtss_phy_ts_eth_conf_t::value`

tag in value/mask

**7.189.2.20 outer\_tag**

`union { ... } vtss_phy_ts_eth_conf_t::outer_tag`

Outer tag

**7.189.2.21 range [2/2]**

```
struct { ... } vtss_phy_ts_eth_conf_t::range
```

tag in range

**7.189.2.22 value [2/2]**

```
struct { ... } vtss_phy_ts_eth_conf_t::value
```

tag in value/mask

**7.189.2.23 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.

**7.189.2.24 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.

**7.189.2.25 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

**7.189.2.26 inner\_tag**

```
union { ... } vtss_phy_ts_eth_conf_t::inner_tag
```

Inner Tag

### 7.189.2.27 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](#)

## 7.190 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](#)

### 7.190.1 Detailed Description

Defines the params for FIFO SYNC function.

Definition at line 2141 of file [vtss\\_phy\\_ts\\_api.h](#).

### 7.190.2 Field Documentation

#### 7.190.2.1 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](#).

#### 7.190.2.2 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.

#### 7.190.2.3 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.

#### 7.190.2.4 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

## 7.191 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 \[6\]](#)

### 7.191.1 Detailed Description

Tx TSFIFO entry signature.

Definition at line 669 of file vtss\_phy\_ts\_api.h.

### 7.191.2 Field Documentation

#### 7.191.2.1 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.

#### 7.191.2.2 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.

#### 7.191.2.3 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.

#### 7.191.2.4 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.

### 7.191.2.5 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`.

### 7.191.2.6 dest\_ip

`u32 vtss_phy_ts_fifo_sig_t::dest_ip`

Destination IP

Definition at line 675 of file `vtss_phy_ts_api.h`.

### 7.191.2.7 src\_ip

`u32 vtss_phy_ts_fifo_sig_t::src_ip`

Source IP

Definition at line 676 of file `vtss_phy_ts_api.h`.

### 7.191.2.8 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`

## 7.192 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]

### 7.192.1 Detailed Description

Analyzer Generic data configuration options using IP comparator.

Definition at line 1122 of file vtss\_phy\_ts\_api.h.

### 7.192.2 Field Documentation

#### 7.192.2.1 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.

#### 7.192.2.2 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.

#### 7.192.2.3 comm\_opt

`struct { ... } vtss_phy_ts_gen_conf_t::comm_opt`

Generic Matching common configuration

#### 7.192.2.4 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`.

#### 7.192.2.5 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`.

#### 7.192.2.6 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`.

#### 7.192.2.7 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`

### 7.193 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`

### 7.193.1 Detailed Description

Generic Action configuration option.

Definition at line 1400 of file `vtss_phy_ts_api.h`.

### 7.193.2 Field Documentation

#### 7.193.2.1 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`.

#### 7.193.2.2 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`.

#### 7.193.2.3 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`.

#### 7.193.2.4 data

`u32 vtss_phy_ts_generic_action_t::data[2]`

Matching data pattern

Definition at line 1404 of file `vtss_phy_ts_api.h`.

#### 7.193.2.5 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`.

#### 7.193.2.6 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`.

#### 7.193.2.7 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`

## 7.194 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

### 7.194.1 Detailed Description

Generic engine flow configuration options.

Definition at line 1168 of file vtss\_phy\_ts\_api.h.

### 7.194.2 Field Documentation

#### 7.194.2.1 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.

#### 7.194.2.2 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](#)

## 7.195 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

### 7.195.1 Detailed Description

Analyzer OAM comparator, IETF MPLS ACH OAM Packet format configuration options.

Definition at line 1368 of file vtss\_phy\_ts\_api.h.

### 7.195.2 Field Documentation

#### 7.195.2.1 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.

#### 7.195.2.2 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.

#### 7.195.2.3 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`

## 7.196 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`

### 7.196.1 Detailed Description

Defines the initial parameters to be passed to init function.

Definition at line 1743 of file vtss\_phy\_ts\_api.h.

### 7.196.2 Field Documentation

#### 7.196.2.1 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.

#### 7.196.2.2 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.

### 7.196.2.3 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`.

### 7.196.2.4 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`.

### 7.196.2.5 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`.

### 7.196.2.6 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`.

### 7.196.2.7 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`.

**7.196.2.8 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.

**7.196.2.9 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.

**7.196.2.10 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.

**7.196.2.11 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.

**7.196.2.12 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.

### 7.196.2.13 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`.

### 7.196.2.14 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`.

### 7.196.2.15 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`

## 7.197 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]
```

### 7.197.1 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.

### 7.197.2 Field Documentation

#### 7.197.2.1 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.

### 7.197.2.2 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.

### 7.197.2.3 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.

### 7.197.2.4 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.

### 7.197.2.5 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.

### 7.197.2.6 comm\_opt

```
struct { ... } vtss_phy_ts_ip_conf_t::comm_opt
```

IP common config

**7.197.2.7 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`.

**7.197.2.8 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`.

**7.197.2.9 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`.

**7.197.2.10 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`.

**7.197.2.11 ipv4**

`struct { ... } vtss_phy_ts_ip_conf_t::ipv4`

IPv4 Address

### 7.197.2.12 ipv6

```
struct { ... } vtss_phy_ts_ip_conf_t::ipv6
```

IPv6 Mask

### 7.197.2.13 ip\_addr

```
union { ... } vtss_phy_ts_ip_conf_t::ip_addr
```

IPv4/IPv6 address to be matched

### 7.197.2.14 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](#)

## 7.198 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]

### 7.198.1 Detailed Description

Analyzer MPLS comparator configuration options.

Definition at line 1061 of file `vtss_phy_ts_api.h`.

### 7.198.2 Field Documentation

#### 7.198.2.1 `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`.

#### 7.198.2.2 `comm_opt`

`struct { ... } vtss_phy_ts_mpls_conf_t::comm_opt`

MPLS common config

#### 7.198.2.3 `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`.

#### 7.198.2.4 `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`.

### 7.198.2.5 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.

### 7.198.2.6 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.

### 7.198.2.7 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.

### 7.198.2.8 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.

### 7.198.2.9 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.

**7.198.2.10 top\_down**

```
struct { ... } vtss_phy_ts_mpls_conf_t::top_down
```

Top down configuration

**7.198.2.11 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.

**7.198.2.12 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.

**7.198.2.13 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.

**7.198.2.14 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.

**7.198.2.15 bottom\_up**

```
struct { ... } vtss_phy_ts_mpls_conf_t::bottom_up
```

Bottom up configuration

### 7.198.2.16 stack\_level

```
union { ... } vtss_phy_ts_mpls_conf_t::stack_level
```

4 level values; top\_down or bottom\_up depends on stack\_ref\_point

### 7.198.2.17 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](#)

## 7.199 vtss\_phy\_ts\_mpls\_lvl\_rng\_t Struct Reference

MPLS level range.

```
#include <vtss_phy_ts_api.h>
```

### Data Fields

- [u32 lower](#)
- [u32 upper](#)

### 7.199.1 Detailed Description

MPLS level range.

Definition at line 1053 of file [vtss\\_phy\\_ts\\_api.h](#).

### 7.199.2 Field Documentation

#### 7.199.2.1 lower

```
u32 vtss_phy_ts_mpls_lvl_rng_t::lower
```

lower range value

Definition at line 1054 of file [vtss\\_phy\\_ts\\_api.h](#).

### 7.199.2.2 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`

## 7.200 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`

### 7.200.1 Detailed Description

n-phase status

Definition at line 1885 of file vtss\_phy\_ts\_api.h.

### 7.200.2 Field Documentation

#### 7.200.2.1 enable

`BOOL vtss_phy_ts_nphase_status_t::enable`

Enabled status

Definition at line 1886 of file vtss\_phy\_ts\_api.h.

### 7.200.2.2 CALIB\_ERR

`BOOL vtss_phy_ts_nphase_status_t::CALIB_ERR`

Calibration error

Definition at line 1887 of file `vtss_phy_ts_api.h`.

### 7.200.2.3 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`

## 7.201 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`

### 7.201.1 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`.

## 7.201.2 Field Documentation

### 7.201.2.1 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.

### 7.201.2.2 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.

### 7.201.2.3 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.

### 7.201.2.4 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.

### 7.201.2.5 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.

### 7.201.2.6 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`.

### 7.201.2.7 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`

## 7.202 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`

### 7.202.1 Detailed Description

OAM engine flow configuration options.

Definition at line 1158 of file `vtss_phy_ts_api.h`.

### 7.202.2 Field Documentation

#### 7.202.2.1 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`.

#### 7.202.2.2 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`.

#### 7.202.2.3 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`.

#### 7.202.2.4 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`

## 7.203 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`

### 7.203.1 Detailed Description

PPS Configuration.

Definition at line 102 of file `vtss_phy_ts_api.h`.

### 7.203.2 Field Documentation

#### 7.203.2.1 `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`.

#### 7.203.2.2 `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`.

#### 7.203.2.3 `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`

## 7.204 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`

### 7.204.1 Detailed Description

Analyzer PTP comparator configuration options.

Definition at line 1268 of file vtss\_phy\_ts\_api.h.

### 7.204.2 Field Documentation

#### 7.204.2.1 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.

#### 7.204.2.2 val

```
u8 vtss_phy_ts_ptp_conf_t::val
```

PTP domain number value

Definition at line 1272 of file vtss\_phy\_ts\_api.h.

#### 7.204.2.3 mask

```
u8 vtss_phy_ts_ptp_conf_t::mask
```

PTP domain number mask

Definition at line 1273 of file vtss\_phy\_ts\_api.h.

#### 7.204.2.4 value

```
struct { ... } vtss_phy_ts_ptp_conf_t::value
```

specific PTP domain, for don't care set mask as '0'

#### 7.204.2.5 upper

```
u8 vtss_phy_ts_ptp_conf_t::upper
```

Ranger upper value

Definition at line 1276 of file vtss\_phy\_ts\_api.h.

#### 7.204.2.6 lower

```
u8 vtss_phy_ts_ptp_conf_t::lower
```

Range lower value

Definition at line 1277 of file vtss\_phy\_ts\_api.h.

#### 7.204.2.7 range

```
struct { ... } vtss_phy_ts_ptp_conf_t::range
```

PTP domain range configuration

#### 7.204.2.8 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

## 7.205 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`

### 7.205.1 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.

### 7.205.2 Field Documentation

#### 7.205.2.1 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.

#### 7.205.2.2 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.

### 7.205.2.3 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`.

### 7.205.2.4 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`.

### 7.205.2.5 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`.

### 7.205.2.6 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`

## 7.206 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

### 7.206.1 Detailed Description

PTP engine flow configuration options.

Definition at line 1146 of file vtss\_phy\_ts\_api.h.

### 7.206.2 Field Documentation

#### 7.206.2.1 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.

#### 7.206.2.2 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.

#### 7.206.2.3 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.

#### 7.206.2.4 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`.

#### 7.206.2.5 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`.

#### 7.206.2.6 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`

### 7.207 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`

#### 7.207.1 Detailed Description

PHY timestamp in seconds and nanoseconds (10 bytes Timestamp)

Definition at line 330 of file `vtss_phy_ts_api.h`.

## 7.207.2 Field Documentation

### 7.207.2.1 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.

### 7.207.2.2 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.

### 7.207.2.3 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

## 7.208 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`

### 7.208.1 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.

### 7.208.2 Field Documentation

#### 7.208.2.1 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.

#### 7.208.2.2 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.

#### 7.208.2.3 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.

#### 7.208.2.4 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.

#### 7.208.2.5 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.

#### 7.208.2.6 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.

#### 7.208.2.7 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.

#### 7.208.2.8 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](#)

## 7.209 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`

### 7.209.1 Detailed Description

Analyzer OAM comparator, Y.1731 OAM Packet format configuration options.

Definition at line 1342 of file vtss\_phy\_ts\_api.h.

### 7.209.2 Field Documentation

#### 7.209.2.1 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.

#### 7.209.2.2 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.

### 7.209.2.3 val

`u8 vtss_phy_ts_y1731_oam_conf_t::val`

Value

Definition at line 1347 of file vtss\_phy\_ts\_api.h.

### 7.209.2.4 mask

`u8 vtss_phy_ts_y1731_oam_conf_t::mask`

Mask

Definition at line 1348 of file vtss\_phy\_ts\_api.h.

### 7.209.2.5 value

`struct { ... } vtss_phy_ts_y1731_oam_conf_t::value`

specific MEG Level, for don't care set mask as '0'

### 7.209.2.6 upper

`u8 vtss_phy_ts_y1731_oam_conf_t::upper`

Range Upper value

Definition at line 1351 of file vtss\_phy\_ts\_api.h.

### 7.209.2.7 lower

`u8 vtss_phy_ts_y1731_oam_conf_t::lower`

Range lower value

Definition at line 1352 of file vtss\_phy\_ts\_api.h.

### 7.209.2.8 range

`struct { ... } vtss_phy_ts_y1731_oam_conf_t::range`

Range configuration

### 7.209.2.9 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](#)

## 7.210 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](#)

### 7.210.1 Detailed Description

Phy type information.

Definition at line 143 of file [vtss\\_phy\\_api.h](#).

### 7.210.2 Field Documentation

#### 7.210.2.1 part\_number

```
u16 vtss_phy_type_t::part_number
```

Part number

Definition at line 145 of file [vtss\\_phy\\_api.h](#).

### 7.210.2.2 revision

`u16 vtss_phy_type_t::revision`

Chip revision

Definition at line 146 of file vtss\_phy\_api.h.

### 7.210.2.3 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.

### 7.210.2.4 channel\_id

`u16 vtss_phy_type_t::channel_id`

Channel id

Definition at line 148 of file vtss\_phy\_api.h.

### 7.210.2.5 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.

### 7.210.2.6 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](#)

## 7.211 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]`

#### 7.211.1 Detailed Description

VeriPHY result.

Definition at line 958 of file vtss\_phy\_api.h.

#### 7.211.2 Field Documentation

##### 7.211.2.1 link

```
BOOL vtss_phy_veriphy_result_t::link
```

Link status

Definition at line 959 of file vtss\_phy\_api.h.

##### 7.211.2.2 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.

### 7.211.2.3 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`

## 7.212 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`

### 7.212.1 Detailed Description

Structure for Get/Set Wake-On-LAN configuration.

Definition at line 1680 of file vtss\_phy\_api.h.

### 7.212.2 Field Documentation

#### 7.212.2.1 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.

### 7.212.2.2 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.

### 7.212.2.3 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.

### 7.212.2.4 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.

### 7.212.2.5 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](#)

## 7.213 vtss\_pi\_conf\_t Struct Reference

PI configuration.

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

## Data Fields

- `u32 cs_wait_ns`

### 7.213.1 Detailed Description

PI configuration.

Definition at line 322 of file `vtss_init_api.h`.

### 7.213.2 Field Documentation

#### 7.213.2.1 `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`

## 7.214 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`

### 7.214.1 Detailed Description

Policer Extensions.

Definition at line 198 of file vtss\_qos\_api.h.

### 7.214.2 Field Documentation

#### 7.214.2.1 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.

#### 7.214.2.2 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.

#### 7.214.2.3 unicast

`BOOL vtss_policer_ext_t::unicast`

Unicast frames are policed

Definition at line 207 of file vtss\_qos\_api.h.

#### 7.214.2.4 multicast

`BOOL vtss_policer_ext_t::multicast`

Multicast frames are policed

Definition at line 208 of file vtss\_qos\_api.h.

#### 7.214.2.5 broadcast

`BOOL vtss_policer_ext_t::broadcast`

Broadcast frames are policed

Definition at line 209 of file vtss\_qos\_api.h.

#### 7.214.2.6 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.

#### 7.214.2.7 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.

#### 7.214.2.8 flooded

`BOOL vtss_policer_ext_t::flooded`

Flooded frames are policed

Definition at line 212 of file vtss\_qos\_api.h.

#### 7.214.2.9 learning

`BOOL vtss_policer_ext_t::learning`

Learning frames are policed

Definition at line 223 of file vtss\_qos\_api.h.

#### 7.214.2.10 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.

#### 7.214.2.11 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.

#### 7.214.2.12 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.

#### 7.214.2.13 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.

#### 7.214.2.14 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](#)

## 7.215 vtss\_policer\_t Struct Reference

Policer.

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

### Data Fields

- `vtss_burst_level_t level`
- `vtss_bitrate_t rate`

#### 7.215.1 Detailed Description

Policer.

Definition at line 185 of file `vtss_qos_api.h`.

#### 7.215.2 Field Documentation

##### 7.215.2.1 level

```
vtss_burst_level_t vtss_policer_t::level
```

Burst level

Definition at line 187 of file `vtss_qos_api.h`.

##### 7.215.2.2 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`

## 7.216 vtss\_port\_bridge\_counters\_t Struct Reference

Port bridge counter structure (RFC 4188)

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

## Data Fields

- [vtss\\_port\\_counter\\_t dot1dTpPortInDiscards](#)

### 7.216.1 Detailed Description

Port bridge counter structure (RFC 4188)

Definition at line 201 of file port.h.

### 7.216.2 Field Documentation

#### 7.216.2.1 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](#)

## 7.217 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](#)

### 7.217.1 Detailed Description

Advertisement control data for Clause 37 aneg.

Definition at line 127 of file vtss\_port\_api.h.

## 7.217.2 Field Documentation

### 7.217.2.1 fdx

`BOOL vtss_port_clause_37_adv_t::fdx`

(FD)

Definition at line 129 of file vtss\_port\_api.h.

### 7.217.2.2 hdx

`BOOL vtss_port_clause_37_adv_t::hdx`

(HD)

Definition at line 130 of file vtss\_port\_api.h.

### 7.217.2.3 symmetric\_pause

`BOOL vtss_port_clause_37_adv_t::symmetric_pause`

(PS1)

Definition at line 131 of file vtss\_port\_api.h.

### 7.217.2.4 asymmetric\_pause

`BOOL vtss_port_clause_37_adv_t::asymmetric_pause`

(PS2)

Definition at line 132 of file vtss\_port\_api.h.

### 7.217.2.5 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.

### 7.217.2.6 acknowledge

`BOOL vtss_port_clause_37_adv_t::acknowledge`

(Ack)

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

### 7.217.2.7 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`

## 7.218 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`

### 7.218.1 Detailed Description

Auto-negotiation control parameter struct.

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

### 7.218.2 Field Documentation

### 7.218.2.1 enable

`BOOL vtss_port_clause_37_control_t::enable`

Enable of Autoneg

Definition at line 154 of file vtss\_port\_api.h.

### 7.218.2.2 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`

## 7.219 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`

### 7.219.1 Detailed Description

Port configuration structure.

Definition at line 268 of file vtss\_port\_api.h.

### 7.219.2 Field Documentation

#### 7.219.2.1 if\_type

`vtss_port_interface_t` `vtss_port_conf_t::if_type`

Interface type

Definition at line 270 of file vtss\_port\_api.h.

#### 7.219.2.2 sd\_enable

`BOOL` `vtss_port_conf_t::sd_enable`

Signal detect enable

Definition at line 271 of file vtss\_port\_api.h.

#### 7.219.2.3 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.

#### 7.219.2.4 sd\_internal

`BOOL` `vtss_port_conf_t::sd_internal`

Internal signal detect selection

Definition at line 273 of file vtss\_port\_api.h.

**7.219.2.5 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.

**7.219.2.6 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.

**7.219.2.7 speed**

```
vtss_port_speed_t vtss_port_conf_t::speed
```

Port speed

Definition at line 276 of file vtss\_port\_api.h.

**7.219.2.8 fdx**

```
BOOL vtss_port_conf_t::fdx
```

Full duplex mode

Definition at line 277 of file vtss\_port\_api.h.

**7.219.2.9 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.

### 7.219.2.10 max\_frame\_length

`u32 vtss_port_conf_t::max_frame_length`

Maximum frame length

Definition at line 279 of file vtss\_port\_api.h.

### 7.219.2.11 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.

### 7.219.2.12 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.

### 7.219.2.13 exc\_col\_cont

`BOOL vtss_port_conf_t::exc_col_cont`

Excessive collision continuation

Definition at line 282 of file vtss\_port\_api.h.

### 7.219.2.14 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.

### 7.219.2.15 xaui\_tx\_lane\_flip

`BOOL vtss_port_conf_t::xaui_tx_lane_flip`

XauI Tx lane flip

Definition at line 284 of file vtss\_port\_api.h.

### 7.219.2.16 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.

### 7.219.2.17 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](#)

## 7.220 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\\_etherlike\\_counters\\_t ethernet\\_like](#)
- [vtss\\_port\\_bridge\\_counters\\_t bridge](#)
- [vtss\\_port\\_proprietary\\_counters\\_t prop](#)

### 7.220.1 Detailed Description

Port counter structure.

Definition at line 220 of file port.h.

### 7.220.2 Field Documentation

#### 7.220.2.1 rmon

```
vtss_port_rmon_counters_t vtss_port_counters_t::rmon
```

RMON counters

Definition at line 222 of file port.h.

#### 7.220.2.2 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.

#### 7.220.2.3 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.

#### 7.220.2.4 bridge

```
vtss_port_bridge_counters_t vtss_port_counters_t::bridge
```

Bridge counters

Definition at line 227 of file port.h.

### 7.220.2.5 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`

## 7.221 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`

### 7.221.1 Detailed Description

Ethernet-like Interface counter structure (RFC 3635)

Definition at line 163 of file port.h.

### 7.221.2 Field Documentation

#### 7.221.2.1 dot3StatsAlignmentErrors

```
vtss_port_counter_t vtss_port_ethernet_like_counters_t::dot3StatsAlignmentErrors
```

Rx alignment errors

Definition at line 165 of file port.h.

#### 7.221.2.2 dot3StatsFCSErrors

```
vtss_port_counter_t vtss_port_ethernet_like_counters_t::dot3StatsFCSErrors
```

Rx FCS errors

Definition at line 166 of file port.h.

#### 7.221.2.3 dot3StatsFrameTooLongs

```
vtss_port_counter_t vtss_port_ethernet_like_counters_t::dot3StatsFrameTooLongs
```

Rx too long

Definition at line 167 of file port.h.

#### 7.221.2.4 dot3StatsSymbolErrors

```
vtss_port_counter_t vtss_port_ethernet_like_counters_t::dot3StatsSymbolErrors
```

Rx symbol errors

Definition at line 168 of file port.h.

#### 7.221.2.5 dot3ControlInUnknownOpcodes

```
vtss_port_counter_t vtss_port_ethernet_like_counters_t::dot3ControlInUnknownOpcodes
```

Rx unknown opcodes

Definition at line 169 of file port.h.

### 7.221.2.6 dot3InPauseFrames

`vtss_port_counter_t vtss_port_ethernet_like_counters_t::dot3InPauseFrames`

Rx pause

Definition at line 171 of file port.h.

### 7.221.2.7 dot3StatsSingleCollisionFrames

`vtss_port_counter_t vtss_port_ethernet_like_counters_t::dot3StatsSingleCollisionFrames`

Tx single collisions

Definition at line 174 of file port.h.

### 7.221.2.8 dot3StatsMultipleCollisionFrames

`vtss_port_counter_t vtss_port_ethernet_like_counters_t::dot3StatsMultipleCollisionFrames`

Tx multiple collisions

Definition at line 175 of file port.h.

### 7.221.2.9 dot3StatsDeferredTransmissions

`vtss_port_counter_t vtss_port_ethernet_like_counters_t::dot3StatsDeferredTransmissions`

Tx deferred

Definition at line 176 of file port.h.

### 7.221.2.10 dot3StatsLateCollisions

`vtss_port_counter_t vtss_port_ethernet_like_counters_t::dot3StatsLateCollisions`

Tx late collisions

Definition at line 177 of file port.h.

### 7.221.2.11 dot3StatsExcessiveCollisions

`vtss_port_counter_t vtss_port_ethernet_like_counters_t::dot3StatsExcessiveCollisions`

Tx excessive collisions

Definition at line 178 of file port.h.

### 7.221.2.12 dot3StatsCarrierSenseErrors

`vtss_port_counter_t vtss_port_ethernet_like_counters_t::dot3StatsCarrierSenseErrors`

Tx carrier sense errors

Definition at line 179 of file port.h.

### 7.221.2.13 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`

## 7.222 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]`

### 7.222.1 Detailed Description

Flow control setup.

Definition at line 189 of file vtss\_port\_api.h.

### 7.222.2 Field Documentation

#### 7.222.2.1 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.

#### 7.222.2.2 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.

#### 7.222.2.3 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.

#### 7.222.2.4 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

## 7.223 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`

#### 7.223.1 Detailed Description

Inter frame gap structure.

Definition at line 206 of file vtss\_port\_api.h.

#### 7.223.2 Field Documentation

##### 7.223.2.1 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.

##### 7.223.2.2 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.

### 7.223.2.3 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](#)

## 7.224 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](#)

### 7.224.1 Detailed Description

Interfaces Group counter structure (RFC 2863)

Definition at line 144 of file port.h.

### 7.224.2 Field Documentation

#### 7.224.2.1 ifInOctets

```
vtss_port_counter_t vtss_port_if_group_counters_t::ifInOctets
```

Rx octets

Definition at line 145 of file port.h.

#### 7.224.2.2 ifInUcastPkts

```
vtss_port_counter_t vtss_port_if_group_counters_t::ifInUcastPkts
```

Rx unicasts

Definition at line 146 of file port.h.

#### 7.224.2.3 ifInMulticastPkts

```
vtss_port_counter_t vtss_port_if_group_counters_t::ifInMulticastPkts
```

Rx multicasts

Definition at line 147 of file port.h.

#### 7.224.2.4 ifInBroadcastPkts

```
vtss_port_counter_t vtss_port_if_group_counters_t::ifInBroadcastPkts
```

Rx broadcasts

Definition at line 148 of file port.h.

#### 7.224.2.5 ifInNUcastPkts

```
vtss_port_counter_t vtss_port_if_group_counters_t::ifInNUcastPkts
```

Rx non-unicasts

Definition at line 149 of file port.h.

#### 7.224.2.6 ifInDiscards

`vtss_port_counter_t vtss_port_if_group_counters_t::ifInDiscards`

Rx discards

Definition at line 150 of file port.h.

#### 7.224.2.7 ifInErrors

`vtss_port_counter_t vtss_port_if_group_counters_t::ifInErrors`

Rx errors

Definition at line 151 of file port.h.

#### 7.224.2.8 ifOutOctets

`vtss_port_counter_t vtss_port_if_group_counters_t::ifOutOctets`

Tx octets

Definition at line 153 of file port.h.

#### 7.224.2.9 ifOutUcastPkts

`vtss_port_counter_t vtss_port_if_group_counters_t::ifOutUcastPkts`

Tx unicasts

Definition at line 154 of file port.h.

#### 7.224.2.10 ifOutMulticastPkts

`vtss_port_counter_t vtss_port_if_group_counters_t::ifOutMulticastPkts`

Tx multicasts

Definition at line 155 of file port.h.

#### 7.224.2.11 ifOutBroadcastPkts

`vtss_port_counter_t vtss_port_if_group_counters_t::ifOutBroadcastPkts`

Tx broadcasts

Definition at line 156 of file port.h.

#### 7.224.2.12 ifOutNUcastPkts

`vtss_port_counter_t vtss_port_if_group_counters_t::ifOutNUcastPkts`

Tx non-unicasts

Definition at line 157 of file port.h.

#### 7.224.2.13 ifOutDiscards

`vtss_port_counter_t vtss_port_if_group_counters_t::ifOutDiscards`

Tx discards

Definition at line 158 of file port.h.

#### 7.224.2.14 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`

## 7.225 vtss\_port\_ifh\_t Struct Reference

Port Internal Frame Header structure.

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

## Data Fields

- `BOOL ena_ifh_header`

### 7.225.1 Detailed Description

Port Internal Frame Header structure.

Definition at line 434 of file vtss\_port\_api.h.

### 7.225.2 Field Documentation

#### 7.225.2.1 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`

## 7.226 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`

### 7.226.1 Detailed Description

Port map structure.

Definition at line 72 of file vtss\_port\_api.h.

## 7.226.2 Field Documentation

### 7.226.2.1 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.

### 7.226.2.2 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.

### 7.226.2.3 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.

### 7.226.2.4 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.

### 7.226.2.5 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`

## 7.227 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]`

### 7.227.1 Detailed Description

Port proprietary counter structure.

Definition at line 207 of file `port.h`.

### 7.227.2 Field Documentation

#### 7.227.2.1 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`.

### 7.227.2.2 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

## 7.228 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

## 7.228.1 Detailed Description

RMON counter structure (RFC 2819)

Definition at line 109 of file port.h.

## 7.228.2 Field Documentation

### 7.228.2.1 rx\_etherStatsDropEvents

```
vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsDropEvents
```

Rx drop events

Definition at line 110 of file port.h.

### 7.228.2.2 rx\_etherStatsOctets

```
vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsOctets
```

Rx octets

Definition at line 111 of file port.h.

### 7.228.2.3 rx\_etherStatsPkts

```
vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsPkts
```

Rx packets

Definition at line 112 of file port.h.

### 7.228.2.4 rx\_etherStatsBroadcastPkts

```
vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsBroadcastPkts
```

Rx broadcasts

Definition at line 113 of file port.h.

**7.228.2.5 rx\_etherStatsMulticastPkts**

```
vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsMulticastPkts
```

Rx multicasts

Definition at line 114 of file port.h.

**7.228.2.6 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.

**7.228.2.7 rx\_etherStatsUndersizePkts**

```
vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsUndersizePkts
```

Rx undersize packets

Definition at line 116 of file port.h.

**7.228.2.8 rx\_etherStatsOversizePkts**

```
vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsOversizePkts
```

Rx oversize packets

Definition at line 117 of file port.h.

**7.228.2.9 rx\_etherStatsFragments**

```
vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsFragments
```

Rx fragments

Definition at line 118 of file port.h.

**7.228.2.10 rx\_etherStatsJabbers**

```
vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsJabbers
```

Rx jabbers

Definition at line 119 of file port.h.

**7.228.2.11 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.

**7.228.2.12 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.

**7.228.2.13 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.

**7.228.2.14 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.

**7.228.2.15 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.

**7.228.2.16 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.

**7.228.2.17 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.

**7.228.2.18 tx\_etherStatsDropEvents**

```
vtss_port_counter_t vtss_port_rmon_counters_t::tx_etherStatsDropEvents
```

Tx drop events

Definition at line 128 of file port.h.

**7.228.2.19 tx\_etherStatsOctets**

```
vtss_port_counter_t vtss_port_rmon_counters_t::tx_etherStatsOctets
```

Tx octets

Definition at line 129 of file port.h.

**7.228.2.20 tx\_etherStatsPkts**

```
vtss_port_counter_t vtss_port_rmon_counters_t::tx_etherStatsPkts
```

Tx packets

Definition at line 130 of file port.h.

**7.228.2.21 tx\_etherStatsBroadcastPkts**

```
vtss_port_counter_t vtss_port_rmon_counters_t::tx_etherStatsBroadcastPkts
```

Tx broadcasts

Definition at line 131 of file port.h.

**7.228.2.22 tx\_etherStatsMulticastPkts**

```
vtss_port_counter_t vtss_port_rmon_counters_t::tx_etherStatsMulticastPkts
```

Tx multicasts

Definition at line 132 of file port.h.

**7.228.2.23 tx\_etherStatsCollisions**

```
vtss_port_counter_t vtss_port_rmon_counters_t::tx_etherStatsCollisions
```

Tx collisions

Definition at line 133 of file port.h.

**7.228.2.24 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.

**7.228.2.25 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.

**7.228.2.26 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.

**7.228.2.27 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.

**7.228.2.28 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.

**7.228.2.29 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.

### 7.228.2.30 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](#)

## 7.229 vtss\_port\_serdes\_conf\_t Struct Reference

SFI Serdes configuration.

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

### Data Fields

- [BOOL sfp\\_dac](#)

### 7.229.1 Detailed Description

SFI Serdes configuration.

Definition at line 261 of file vtss\_port\_api.h.

### 7.229.2 Field Documentation

#### 7.229.2.1 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](#)

## 7.230 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`

#### 7.230.1 Detailed Description

Advertisement control data for SGMII aneg.

Definition at line 139 of file vtss\_port\_api.h.

#### 7.230.2 Field Documentation

##### 7.230.2.1 link

```
BOOL vtss_port_sgmii_aneg_t::link
```

LP link status

Definition at line 141 of file vtss\_port\_api.h.

##### 7.230.2.2 fdx

```
BOOL vtss_port_sgmii_aneg_t::fdx
```

FD

Definition at line 142 of file vtss\_port\_api.h.

### 7.230.2.3 hdx

`BOOL vtss_port_sgmii_aneg_t::hdx`

HD

Definition at line 143 of file vtss\_port\_api.h.

### 7.230.2.4 speed\_10M

`BOOL vtss_port_sgmii_aneg_t::speed_10M`

speed 10 advertised

Definition at line 144 of file vtss\_port\_api.h.

### 7.230.2.5 speed\_100M

`BOOL vtss_port_sgmii_aneg_t::speed_100M`

speed 100 advertised

Definition at line 145 of file vtss\_port\_api.h.

### 7.230.2.6 speed\_1G

`BOOL vtss_port_sgmii_aneg_t::speed_1G`

speed 1G advertised

Definition at line 146 of file vtss\_port\_api.h.

### 7.230.2.7 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](#)
-

## 7.231 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`

### 7.231.1 Detailed Description

Port status parameter struct.

Definition at line 295 of file port.h.

### 7.231.2 Field Documentation

#### 7.231.2.1 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.

#### 7.231.2.2 link

```
BOOL vtss_port_status_t::link
```

Link is up. Remaining fields only valid if TRUE

Definition at line 298 of file port.h.

### 7.231.2.3 speed

`vtss_port_speed_t vtss_port_status_t::speed`

Speed

Definition at line 299 of file port.h.

### 7.231.2.4 fdx

`BOOL vtss_port_status_t::fdx`

Full duplex

Definition at line 300 of file port.h.

### 7.231.2.5 remote\_fault

`BOOL vtss_port_status_t::remote_fault`

Remote fault signalled

Definition at line 301 of file port.h.

### 7.231.2.6 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.

### 7.231.2.7 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.

### 7.231.2.8 aneg

`vtss_aneg_t vtss_port_status_t::aneg`

Auto negotiation result

Definition at line 307 of file port.h.

### 7.231.2.9 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.

### 7.231.2.10 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.

### 7.231.2.11 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`

## 7.232 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`

### 7.232.1 Detailed Description

QCE action.

Definition at line 566 of file vtss\_qos\_api.h.

### 7.232.2 Field Documentation

#### 7.232.2.1 prio\_enable

`BOOL vtss_qce_action_t::prio_enable`

Enable priority classification

Definition at line 568 of file vtss\_qos\_api.h.

#### 7.232.2.2 prio

`vtss_prio_t vtss_qce_action_t::prio`

Priority value

Definition at line 569 of file vtss\_qos\_api.h.

#### 7.232.2.3 dp\_enable

`BOOL vtss_qce_action_t::dp_enable`

Enable DP classification

Definition at line 570 of file vtss\_qos\_api.h.

#### 7.232.2.4 dp

`vtss_dp_level_t vtss_qce_action_t::dp`

DP value

Definition at line 571 of file `vtss_qos_api.h`.

#### 7.232.2.5 dscp\_enable

`BOOL vtss_qce_action_t::dscp_enable`

Enable DSCP classification

Definition at line 572 of file `vtss_qos_api.h`.

#### 7.232.2.6 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`

### 7.233 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`

#### 7.233.1 Detailed Description

Frame data for VTSS\_QCE\_TYPE\_ETYPE.

Definition at line 494 of file `vtss_qos_api.h`.

## 7.233.2 Field Documentation

### 7.233.2.1 etype

`vtss_vcap_u16_t` `vtss_qce_frame_etype_t::etype`

Ethernet Type value

Definition at line 496 of file `vtss_qos_api.h`.

### 7.233.2.2 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`

## 7.234 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`

### 7.234.1 Detailed Description

Frame data for VTSS\_QCE\_TYPE\_IPV4.

Definition at line 513 of file `vtss_qos_api.h`.

## 7.234.2 Field Documentation

### 7.234.2.1 fragment

`vtss_vcap_bit_t` `vtss_qce_frame_ipv4_t::fragment`

Fragment

Definition at line 515 of file vtss\_qos\_api.h.

### 7.234.2.2 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.

### 7.234.2.3 proto

`vtss_vcap_u8_t` `vtss_qce_frame_ipv4_t::proto`

Protocol

Definition at line 517 of file vtss\_qos\_api.h.

### 7.234.2.4 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.

### 7.234.2.5 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.

#### 7.234.2.6 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`

## 7.235 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`

### 7.235.1 Detailed Description

Frame data for VTSS\_QCE\_TYPE\_IPV6.

Definition at line 527 of file vtss\_qos\_api.h.

### 7.235.2 Field Documentation

#### 7.235.2.1 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.

### 7.235.2.2 proto

`vtss_vcap_u8_t vtss_qce_frame_ipv6_t::proto`

Protocol

Definition at line 530 of file `vtss_qos_api.h`.

### 7.235.2.3 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`.

### 7.235.2.4 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`.

### 7.235.2.5 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`

## 7.236 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](#)

### 7.236.1 Detailed Description

Frame data for VTSS\_QCE\_TYPE\_LLC.

Definition at line 501 of file [vtss\\_qos\\_api.h](#).

### 7.236.2 Field Documentation

#### 7.236.2.1 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](#)

## 7.237 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](#)

### 7.237.1 Detailed Description

Frame data for VTSS\_QCE\_TYPE\_SNAP.

Definition at line 507 of file [vtss\\_qos\\_api.h](#).

### 7.237.2 Field Documentation

### 7.237.2.1 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](#)

## 7.238 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`

### 7.238.1 Detailed Description

QCE key.

Definition at line 542 of file vtss\_qos\_api.h.

### 7.238.2 Field Documentation

#### 7.238.2.1 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.

### 7.238.2.2 mac

`vtss_qce_mac_t` `vtss_qce_key_t::mac`

MAC

Definition at line 545 of file vtss\_qos\_api.h.

### 7.238.2.3 tag

`vtss_qce_tag_t` `vtss_qce_key_t::tag`

Tag

Definition at line 546 of file vtss\_qos\_api.h.

### 7.238.2.4 type

`vtss_qce_type_t` `vtss_qce_key_t::type`

Frame type

Definition at line 550 of file vtss\_qos\_api.h.

### 7.238.2.5 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.

### 7.238.2.6 llc

`vtss_qce_frame_llc_t` `vtss_qce_key_t::llc`

VTSS\_QCE\_TYPE\_LLCA

Definition at line 556 of file vtss\_qos\_api.h.

### 7.238.2.7 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`.

### 7.238.2.8 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`.

### 7.238.2.9 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`.

### 7.238.2.10 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`

## 7.239 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`

### 7.239.1 Detailed Description

QCE MAC information.

Definition at line 471 of file `vtss_qos_api.h`.

### 7.239.2 Field Documentation

#### 7.239.2.1 dmac\_mc

`vtss_vcap_bit_t vtss_qce_mac_t::dmac_mc`

Multicast DMAC

Definition at line 473 of file `vtss_qos_api.h`.

#### 7.239.2.2 dmac\_bc

`vtss_vcap_bit_t vtss_qce_mac_t::dmac_bc`

Broadcast DMAC

Definition at line 474 of file `vtss_qos_api.h`.

#### 7.239.2.3 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`

## 7.240 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`

#### 7.240.1 Detailed Description

QoS Control Entry.

Definition at line 588 of file vtss\_qos\_api.h.

#### 7.240.2 Field Documentation

##### 7.240.2.1 id

```
vtss_qce_id_t vtss_qce_t::id
```

Entry ID

Definition at line 590 of file vtss\_qos\_api.h.

##### 7.240.2.2 key

```
vtss_qce_key_t vtss_qce_t::key
```

QCE key

Definition at line 591 of file vtss\_qos\_api.h.

### 7.240.2.3 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`

## 7.241 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`

### 7.241.1 Detailed Description

QCE tag information.

Definition at line 482 of file vtss\_qos\_api.h.

### 7.241.2 Field Documentation

#### 7.241.2.1 vid

`vtss_vcap_vr_t` `vtss_qce_tag_t::vid`

VLAN ID (12 bit)

Definition at line 484 of file vtss\_qos\_api.h.

### 7.241.2.2 pcp

`vtss_vcap_u8_t vtss_qce_tag_t::pcp`

PCP (3 bit)

Definition at line 485 of file `vtss_qos_api.h`.

### 7.241.2.3 dei

`vtss_vcap_bit_t vtss_qce_tag_t::dei`

DEI

Definition at line 486 of file `vtss_qos_api.h`.

### 7.241.2.4 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`

## 7.242 `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]`

### 7.242.1 Detailed Description

All parameters below are defined per chip.

Definition at line 91 of file vtss\_qos\_api.h.

### 7.242.2 Field Documentation

#### 7.242.2.1 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.

#### 7.242.2.2 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.

#### 7.242.2.3 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.

#### 7.242.2.4 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.

#### 7.242.2.5 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.

#### 7.242.2.6 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.

#### 7.242.2.7 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.

#### 7.242.2.8 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](#)

### 7.243 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]`

### 7.243.1 Detailed Description

QoS setup per port.

Definition at line 344 of file vtss\_qos\_api.h.

### 7.243.2 Field Documentation

#### 7.243.2.1 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.

### 7.243.2.2 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.

### 7.243.2.3 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.

### 7.243.2.4 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.

### 7.243.2.5 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.

### 7.243.2.6 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.

### 7.243.2.7 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.

### 7.243.2.8 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.

### 7.243.2.9 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.

### 7.243.2.10 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.

### 7.243.2.11 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.

### 7.243.2.12 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.

### 7.243.2.13 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.

### 7.243.2.14 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.

### 7.243.2.15 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.

### 7.243.2.16 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.

### 7.243.2.17 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.

### 7.243.2.18 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.

### 7.243.2.19 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.

### 7.243.2.20 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.

### 7.243.2.21 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.

### 7.243.2.22 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.

### 7.243.2.23 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.

### 7.243.2.24 dwrr\_enable

```
BOOL vtss_qos_port_conf_t::dwrr_enable
```

Enable Weighted fairness queueing

Definition at line 400 of file vtss\_qos\_api.h.

### 7.243.2.25 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

## 7.244 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.*

### 7.244.1 Detailed Description

Queue System settings.

Definition at line 399 of file vtss\_init\_api.h.

### 7.244.2 Field Documentation

#### 7.244.2.1 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.

#### 7.244.2.2 oversubscription

`vtss_pct_t vtss_qs_conf_t::oversubscription`

Queue System oversubscription 0-50%.

Definition at line 401 of file vtss\_init\_api.h.

#### 7.244.2.3 port\_min

`u32 vtss_qs_conf_t::port_min`

Minumum Guaranteed port buffer. Bytes.

Definition at line 405 of file vtss\_init\_api.h.

#### 7.244.2.4 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.

#### 7.244.2.5 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.

#### 7.244.2.6 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.

#### 7.244.2.7 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](#)

### 7.245 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`

### 7.245.1 Detailed Description

Structure for Get PHY RC-PLL status.

Definition at line 1742 of file `vtss_phy_api.h`.

### 7.245.2 Field Documentation

#### 7.245.2.1 `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`.

#### 7.245.2.2 `cal_error`

```
u8 vtss_rcpll_status_t::cal_error
```

Calibration Error indication

Definition at line 1744 of file `vtss_phy_api.h`.

#### 7.245.2.3 `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`

## 7.246 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`

#### 7.246.1 Detailed Description

Random Early Detection configuration struct version 1 (per port, per queue)

Definition at line 48 of file vtss\_qos\_api.h.

#### 7.246.2 Field Documentation

##### 7.246.2.1 enable

```
BOOL vtss_red_t::enable
```

Enable/disable RED

Definition at line 50 of file vtss\_qos\_api.h.

##### 7.246.2.2 max\_th

```
vtss_pct_t vtss_red_t::max_th
```

Maximum threshold

Definition at line 52 of file vtss\_qos\_api.h.

## 7.246.2.3 min\_th

```
vtss_pct_t vtss_red_t::min_th
```

Minimum threshold

Definition at line 53 of file vtss\_qos\_api.h.

## 7.246.2.4 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.

## 7.246.2.5 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.

## 7.246.2.6 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](#)

## 7.247 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`

### 7.247.1 Detailed Description

Restart status.

Definition at line 608 of file `vtss_init_api.h`.

### 7.247.2 Field Documentation

#### 7.247.2.1 restart

`vtss_restart_t vtss_restart_status_t::restart`

Previous restart mode

Definition at line 609 of file `vtss_init_api.h`.

#### 7.247.2.2 prev\_version

`vtss_version_t vtss_restart_status_t::prev_version`

Previous API version

Definition at line 610 of file `vtss_init_api.h`.

#### 7.247.2.3 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`

## 7.248 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`

### 7.248.1 Detailed Description

Routing entry.

Definition at line 868 of file types.h.

### 7.248.2 Field Documentation

#### 7.248.2.1 type

```
vtss_routing_entry_type_t vtss_routing_entry_t::type
```

Type of route

Definition at line 871 of file types.h.

#### 7.248.2.2 ipv4\_uc

```
vtss_ipv4_uc_t vtss_routing_entry_t::ipv4_uc
```

IPv6 unicast route

Definition at line 875 of file types.h.

### 7.248.2.3 ipv6\_uc

`vtss_ipv6_uc_t vtss_routing_entry_t::ipv6_uc`

IPv6 unicast route

Definition at line 878 of file types.h.

### 7.248.2.4 route

`union { ... } vtss_routing_entry_t::route`

Route

### 7.248.2.5 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`

## 7.249 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]`

### 7.249.1 Detailed Description

Structure for Wake-On-LAN Secure-On Password.

Definition at line 1664 of file vtss\_phy\_api.h.

## 7.249.2 Field Documentation

### 7.249.2.1 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`

## 7.250 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`

### 7.250.1 Detailed Description

Serdes macro configuration.

Definition at line 363 of file vtss\_init\_api.h.

## 7.250.2 Field Documentation

### 7.250.2.1 serdes1g\_vdd

`vtss_vdd_t vtss_serdes_macro_conf_t::serdes1g_vdd`

Serdes1g supply

Definition at line 364 of file vtss\_init\_api.h.

### 7.250.2.2 serdes6g\_vdd

`vtss_vdd_t` `vtss_serdes_macro_conf_t::serdes6g_vdd`

Serdes6g supply

Definition at line 365 of file `vtss_init_api.h`.

### 7.250.2.3 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`.

### 7.250.2.4 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`

## 7.251 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`

### 7.251.1 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.

### 7.251.2 Field Documentation

#### 7.251.2.1 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.

#### 7.251.2.2 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](#)

## 7.252 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 \[VTSS\\_SGPIO\\_PORTS\]](#)

### 7.252.1 Detailed Description

SGPIO configuration for a group.

Definition at line 797 of file vtss\_misc\_api.h.

### 7.252.2 Field Documentation

#### 7.252.2.1 bmode

```
vtss_sgpio_bmode_t vtss_sgpio_conf_t::bmode[2]
```

Blink mode 0 and 1

Definition at line 799 of file vtss\_misc\_api.h.

#### 7.252.2.2 bit\_count

```
u8 vtss_sgpio_conf_t::bit_count
```

Bits enabled per port, 1-4

Definition at line 800 of file vtss\_misc\_api.h.

#### 7.252.2.3 port\_conf

```
vtss_sgpio_port_conf_t vtss_sgpio_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

## 7.253 vtss\_sgpio\_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]`

### 7.253.1 Detailed Description

SGPIO port configuration.

Definition at line 789 of file vtss\_misc\_api.h.

### 7.253.2 Field Documentation

#### 7.253.2.1 enabled

`BOOL vtss_sgpi_port_conf_t::enabled`

Port enabled/disabled

Definition at line 791 of file vtss\_misc\_api.h.

#### 7.253.2.2 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.

#### 7.253.2.3 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](#)

## 7.254 vtss\_sgpi\_port\_data\_t Struct Reference

SGPIO read data for a port.

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

### Data Fields

- `BOOL value [4]`

#### 7.254.1 Detailed Description

SGPIO read data for a port.

Definition at line 835 of file `vtss_misc_api.h`.

#### 7.254.2 Field Documentation

##### 7.254.2.1 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`

## 7.255 vtss\_shaper\_t Struct Reference

Shaper.

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

### Data Fields

- `vtss_burst_level_t level`
- `vtss_bitrate_t rate`

### 7.255.1 Detailed Description

Shaper.

Definition at line 298 of file vtss\_qos\_api.h.

### 7.255.2 Field Documentation

#### 7.255.2.1 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.

#### 7.255.2.2 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](#)

## 7.256 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`

### 7.256.1 Detailed Description

10G Phy link and fault status

Definition at line 86 of file vtss\_phy\_10g\_api.h.

### 7.256.2 Field Documentation

#### 7.256.2.1 rx\_link

```
BOOL vtss_sublayer_status_t::rx_link
```

The rx link status

Definition at line 87 of file vtss\_phy\_10g\_api.h.

#### 7.256.2.2 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.

#### 7.256.2.3 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.

#### 7.256.2.4 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](#)

## 7.257 `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`

#### 7.257.1 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`.

#### 7.257.2 Field Documentation

##### 7.257.2.1 `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`.

##### 7.257.2.2 `squelsh`

```
BOOL vtss_sync_clock_in_t::squelsh
```

Enable/disable of automatic squelsh

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

### 7.257.2.3 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`

## 7.258 `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_clock_out_t divider`
- `BOOL enable`

### 7.258.1 Detailed Description

Struct containing configuration for a recovered clock output port.

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

### 7.258.2 Field Documentation

#### 7.258.2.1 divider

`vtss_sync_clock_out_t::divider`

Selection the clock division. This should be set to `VTSS_SYNC_CLOCK_OUT_DIVIDER_1` if recovered clock is comming from internal PHY

Definition at line 75 of file `vtss_sync_api.h`.

### 7.258.2.2 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`

## 7.259 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`

### 7.259.1 Detailed Description

Tag Control Information (according to IEEE 802.1Q)

Definition at line 206 of file vtss\_packet\_api.h.

### 7.259.2 Field Documentation

#### 7.259.2.1 vid

`vtss_vid_t vtss_tci_t::vid`

VLAN ID

Definition at line 208 of file vtss\_packet\_api.h.

### 7.259.2.2 cfi

`BOOL vtss_tci_t::cfi`

Canonical Format Indicator

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

### 7.259.2.3 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`

## 7.260 vtss\_timeofday\_t Struct Reference

Time of day structure.

```
#include <vtss_os_ecos.h>
```

### Data Fields

- `u32 sec`
- `time_t sec`

### 7.260.1 Detailed Description

Time of day structure.

Definition at line 59 of file `vtss_os_ecos.h`.

### 7.260.2 Field Documentation

### 7.260.2.1 sec [1/2]

`u32 vtss_timeofday_t::sec`

Time of day in seconds

Definition at line 60 of file vtss\_os\_ecos.h.

### 7.260.2.2 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`

## 7.261 vtss\_timestamp\_t Struct Reference

Time stamp in seconds and nanoseconds.

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

### Data Fields

- `u16 sec_msb`
- `u32 seconds`
- `u32 nanoseconds`

### 7.261.1 Detailed Description

Time stamp in seconds and nanoseconds.

Definition at line 1212 of file types.h.

### 7.261.2 Field Documentation

### 7.261.2.1 sec\_msb

`u16 vtss_timestamp_t::sec_msb`

Seconds msb

Definition at line 1213 of file types.h.

### 7.261.2.2 seconds

`u32 vtss_timestamp_t::seconds`

Seconds

Definition at line 1214 of file types.h.

### 7.261.2.3 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`

## 7.262 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]`

### 7.262.1 Detailed Description

Trace group configuration.

Definition at line 97 of file vtss\_misc\_api.h.

## 7.262.2 Field Documentation

### 7.262.2.1 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

## 7.263 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](#)

### 7.263.1 Detailed Description

external clock output configuration.

Definition at line 289 of file vtss\_ts\_api.h.

## 7.263.2 Field Documentation

### 7.263.2.1 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.

### 7.263.2.2 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`.

### 7.263.2.3 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`

## 7.264 `vtss_ts_id_t` Struct Reference

Timestamp identifier.

```
#include <vtss_ts_api.h>
```

### Data Fields

- `u32 ts_id`

### 7.264.1 Detailed Description

Timestamp identifier.

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

### 7.264.2 Field Documentation

### 7.264.2.1 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](#)

## 7.265 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](#)

### 7.265.1 Detailed Description

Hardware timestamping format mode for internal ports.

Definition at line 498 of file vtss\_ts\_api.h.

### 7.265.2 Field Documentation

#### 7.265.2.1 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](#)

## 7.266 vtss\_ts\_operation\_mode\_t Struct Reference

Timestamp operation.

```
#include <vtss_ts_api.h>
```

### Data Fields

- `vtss_ts_mode_t mode`

#### 7.266.1 Detailed Description

Timestamp operation.

Definition at line 451 of file `vtss_ts_api.h`.

#### 7.266.2 Field Documentation

##### 7.266.2.1 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`

## 7.267 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)`

### 7.267.1 Detailed Description

Timestamp allocation.

Definition at line 613 of file vtss\_ts\_api.h.

### 7.267.2 Field Documentation

#### 7.267.2.1 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.

#### 7.267.2.2 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.

#### 7.267.2.3 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

## 7.268 vtss\_ts\_timestamp\_t Struct Reference

Timestamp structure.

```
#include <vtss_ts_api.h>
```

## Data Fields

- `u32 ts`
- `u32 id`
- `void * context`
- `BOOL ts_valid`

### 7.268.1 Detailed Description

Timestamp structure.

Definition at line 532 of file vtss\_ts\_api.h.

### 7.268.2 Field Documentation

#### 7.268.2.1 ts

`u32 vtss_ts_timestamp_t::ts`

Timestamp value

Definition at line 533 of file vtss\_ts\_api.h.

#### 7.268.2.2 id

`u32 vtss_ts_timestamp_t::id`

Timestamp identifier

Definition at line 534 of file vtss\_ts\_api.h.

#### 7.268.2.3 context

`void* vtss_ts_timestamp_t::context`

Application specific context

Definition at line 535 of file vtss\_ts\_api.h.

### 7.268.2.4 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`

## 7.269 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`

### 7.269.1 Detailed Description

VCAP IPv4 address value and mask.

Definition at line 968 of file types.h.

### 7.269.2 Field Documentation

#### 7.269.2.1 value

`vtss_ip_t vtss_vcap_ip_t::value`

Value

Definition at line 970 of file types.h.

### 7.269.2.2 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`

## 7.270 vtss\_vcap\_u128\_t Struct Reference

VCAP 128 bit value and mask.

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

### Data Fields

- `u8 value [16]`
- `u8 mask [16]`

### 7.270.1 Detailed Description

VCAP 128 bit value and mask.

Definition at line 954 of file types.h.

### 7.270.2 Field Documentation

#### 7.270.2.1 value

`u8 vtss_vcap_u128_t::value[16]`

Value

Definition at line 956 of file types.h.

### 7.270.2.2 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](#)

## 7.271 vtss\_vcap\_u16\_t Struct Reference

VCAP 16 bit value and mask.

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

### Data Fields

- [u8 value \[2\]](#)
- [u8 mask \[2\]](#)

### 7.271.1 Detailed Description

VCAP 16 bit value and mask.

Definition at line 919 of file types.h.

### 7.271.2 Field Documentation

#### 7.271.2.1 value

`u8 vtss_vcap_u16_t::value[2]`

Value

Definition at line 921 of file types.h.

### 7.271.2.2 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](#)

## 7.272 vtss\_vcap\_u24\_t Struct Reference

VCAP 24 bit value and mask.

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

### Data Fields

- [u8 value \[3\]](#)
- [u8 mask \[3\]](#)

### 7.272.1 Detailed Description

VCAP 24 bit value and mask.

Definition at line 926 of file types.h.

### 7.272.2 Field Documentation

#### 7.272.2.1 value

`u8 vtss_vcap_u24_t::value[3]`

Value

Definition at line 928 of file types.h.

### 7.272.2.2 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](#)

## 7.273 vtss\_vcap\_u32\_t Struct Reference

VCAP 32 bit value and mask.

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

### Data Fields

- [u8 value \[4\]](#)
- [u8 mask \[4\]](#)

### 7.273.1 Detailed Description

VCAP 32 bit value and mask.

Definition at line 933 of file types.h.

### 7.273.2 Field Documentation

#### 7.273.2.1 value

`u8 vtss_vcap_u32_t::value [4]`

Value

Definition at line 935 of file types.h.

### 7.273.2.2 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](#)

## 7.274 vtss\_vcap\_u40\_t Struct Reference

VCAP 40 bit value and mask.

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

### Data Fields

- [u8 value \[5\]](#)
- [u8 mask \[5\]](#)

### 7.274.1 Detailed Description

VCAP 40 bit value and mask.

Definition at line 940 of file types.h.

### 7.274.2 Field Documentation

#### 7.274.2.1 value

`u8 vtss_vcap_u40_t::value[5]`

Value

Definition at line 942 of file types.h.

### 7.274.2.2 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](#)

## 7.275 vtss\_vcap\_u48\_t Struct Reference

VCAP 48 bit value and mask.

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

### Data Fields

- [u8 value \[6\]](#)
- [u8 mask \[6\]](#)

### 7.275.1 Detailed Description

VCAP 48 bit value and mask.

Definition at line 947 of file types.h.

### 7.275.2 Field Documentation

#### 7.275.2.1 value

`u8 vtss_vcap_u48_t::value[6]`

Value

Definition at line 949 of file types.h.

### 7.275.2.2 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](#)

## 7.276 vtss\_vcap\_u8\_t Struct Reference

VCAP 8 bit value and mask.

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

### Data Fields

- [u8 value](#)
- [u8 mask](#)

### 7.276.1 Detailed Description

VCAP 8 bit value and mask.

Definition at line 912 of file types.h.

### 7.276.2 Field Documentation

#### 7.276.2.1 value

`u8 vtss_vcap_u8_t::value`

Value

Definition at line 914 of file types.h.

### 7.276.2.2 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](#)

## 7.277 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](#)

### 7.277.1 Detailed Description

VCAP UDP/TCP port range.

Definition at line 975 of file types.h.

### 7.277.2 Field Documentation

#### 7.277.2.1 in\_range

`BOOL vtss_vcap_udp_tcp_t::in_range`

Port in range match

Definition at line 977 of file types.h.

### 7.277.2.2 low

`vtss_udp_tcp_t vtss_vcap_udp_tcp_t::low`

Port low value

Definition at line 978 of file types.h.

### 7.277.2.3 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`

## 7.278 vtss\_vcap\_vid\_t Struct Reference

VCAP VLAN ID value and mask.

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

### Data Fields

- `u16 value`
- `u16 mask`

### 7.278.1 Detailed Description

VCAP VLAN ID value and mask.

Definition at line 961 of file types.h.

### 7.278.2 Field Documentation

### 7.278.2.1 value

`u16 vtss_vcap_vid_t::value`

Value

Definition at line 963 of file types.h.

### 7.278.2.2 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](#)

## 7.279 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`

### 7.279.1 Detailed Description

VCAP universal value or range.

Definition at line 994 of file types.h.

## 7.279.2 Field Documentation

### 7.279.2.1 type

`vtss_vcap_vr_type_t` `vtss_vcap_vr_t::type`

Type

Definition at line 996 of file types.h.

### 7.279.2.2 value

`vtss_vcap_vr_value_t` `vtss_vcap_vr_t::value`

Value

Definition at line 1001 of file types.h.

### 7.279.2.3 mask

`vtss_vcap_vr_value_t` `vtss_vcap_vr_t::mask`

Mask, cleared bits are wildcards

Definition at line 1002 of file types.h.

### 7.279.2.4 v

`struct { ... }` `vtss_vcap_vr_t::v`

`type == VTSS_VCAP_VR_TYPE_VALUE_MASK`

### 7.279.2.5 low

`vtss_vcap_vr_value_t` `vtss_vcap_vr_t::low`

Low value

Definition at line 1006 of file types.h.

### 7.279.2.6 high

```
vtss_vcap_vr_value_t vtss_vcap_vr_t::high
```

High value

Definition at line 1007 of file types.h.

### 7.279.2.7 r

```
struct { ... } vtss_vcap_vr_t::r  
type == VTSS_VCAP_VR_TYPE_RANGE_XXXXXX
```

### 7.279.2.8 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](#)

## 7.280 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](#)

### 7.280.1 Detailed Description

VCE Action.

Definition at line 1006 of file vtss\_l2\_api.h.

### 7.280.2 Field Documentation

### 7.280.2.1 vid

```
vtss_vid_t vtss_vce_action_t::vid
```

Classified VLAN ID

Definition at line 1008 of file vtss\_l2\_api.h.

### 7.280.2.2 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](#)

## 7.281 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

### 7.281.1 Detailed Description

Frame data for VTSS\_VCE\_TYPE\_ETYPE.

Definition at line 948 of file vtss\_l2\_api.h.

### 7.281.2 Field Documentation

### 7.281.2.1 etype

`vtss_vcap_u16_t vtss_vce_frame_etype_t::etype`

Ethernet Type value

Definition at line 950 of file vtss\_l2\_api.h.

### 7.281.2.2 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`

## 7.282 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`

### 7.282.1 Detailed Description

Frame data for VTSS\_VCE\_TYPE\_IPV4.

Definition at line 967 of file vtss\_l2\_api.h.

### 7.282.2 Field Documentation

### 7.282.2.1 fragment

`vtss_vcap_bit_t` `vtss_vce_frame_ipv4_t::fragment`

Fragment

Definition at line 969 of file vtss\_l2\_api.h.

### 7.282.2.2 options

`vtss_vcap_bit_t` `vtss_vce_frame_ipv4_t::options`

Header options

Definition at line 970 of file vtss\_l2\_api.h.

### 7.282.2.3 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.

### 7.282.2.4 proto

`vtss_vcap_u8_t` `vtss_vce_frame_ipv4_t::proto`

Protocol

Definition at line 972 of file vtss\_l2\_api.h.

### 7.282.2.5 sip

`vtss_vcap_ip_t` `vtss_vce_frame_ipv4_t::sip`

Source IP address

Definition at line 973 of file vtss\_l2\_api.h.

### 7.282.2.6 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`

## 7.283 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`

### 7.283.1 Detailed Description

Frame data for VTSS\_VCE\_TYPE\_IPV6.

Definition at line 978 of file vtss\_l2\_api.h.

### 7.283.2 Field Documentation

#### 7.283.2.1 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.

### 7.283.2.2 proto

`vtss_vcap_u8_t` `vtss_vce_frame_ipv6_t::proto`

Protocol

Definition at line 981 of file vtss\_l2\_api.h.

### 7.283.2.3 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.

### 7.283.2.4 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`

## 7.284 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`

### 7.284.1 Detailed Description

Frame data for VTSS\_VCE\_TYPE LLC.

Definition at line 955 of file vtss\_l2\_api.h.

## 7.284.2 Field Documentation

### 7.284.2.1 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`

## 7.285 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`

### 7.285.1 Detailed Description

Frame data for VTSS\_VCE\_TYPE\_SNAP.

Definition at line 961 of file vtss\_l2\_api.h.

## 7.285.2 Field Documentation

### 7.285.2.1 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`

## 7.286 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`}

### 7.286.1 Detailed Description

VCE Key.

Definition at line 987 of file vtss\_l2\_api.h.

### 7.286.2 Field Documentation

#### 7.286.2.1 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.

#### 7.286.2.2 mac

```
vtss_vce_mac_t vtss_vce_key_t::mac
```

MAC header

Definition at line 990 of file vtss\_l2\_api.h.

### 7.286.2.3 tag

`vtss_vce_tag_t` `vtss_vce_key_t::tag`

Tag

Definition at line 991 of file vtss\_l2\_api.h.

### 7.286.2.4 type

`vtss_vce_type_t` `vtss_vce_key_t::type`

VCE frame type

Definition at line 992 of file vtss\_l2\_api.h.

### 7.286.2.5 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.

### 7.286.2.6 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.

### 7.286.2.7 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.

### 7.286.2.8 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.

### 7.286.2.9 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.

### 7.286.2.10 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`

## 7.287 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`

### 7.287.1 Detailed Description

VCE MAC header information.

Definition at line 930 of file vtss\_l2\_api.h.

## 7.287.2 Field Documentation

### 7.287.2.1 dmac\_mc

`vtss_vcap_bit_t` `vtss_vce_mac_t::dmac_mc`

Multicast DMAC

Definition at line 932 of file vtss\_l2\_api.h.

### 7.287.2.2 dmac\_bc

`vtss_vcap_bit_t` `vtss_vce_mac_t::dmac_bc`

Broadcast DMAC

Definition at line 933 of file vtss\_l2\_api.h.

### 7.287.2.3 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`

## 7.288 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`

### 7.288.1 Detailed Description

VLAN Control Entry.

Definition at line 1013 of file vtss\_l2\_api.h.

### 7.288.2 Field Documentation

#### 7.288.2.1 id

`vtss_vce_id_t` vtss\_vce\_t::id

VCE ID

Definition at line 1015 of file vtss\_l2\_api.h.

#### 7.288.2.2 key

`vtss_vce_key_t` vtss\_vce\_t::key

VCE Key

Definition at line 1016 of file vtss\_l2\_api.h.

#### 7.288.2.3 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](#)

## 7.289 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`

### 7.289.1 Detailed Description

VCE tag information.

Definition at line 938 of file vtss\_l2\_api.h.

### 7.289.2 Field Documentation

#### 7.289.2.1 vid

`vtss_vcap_vid_t vtss_vce_tag_t::vid`

VLAN ID (12 bit)

Definition at line 940 of file vtss\_l2\_api.h.

#### 7.289.2.2 pcp

`vtss_vcap_u8_t vtss_vce_tag_t::pcp`

PCP (3 bit)

Definition at line 941 of file vtss\_l2\_api.h.

#### 7.289.2.3 dei

`vtss_vcap_bit_t vtss_vce_tag_t::dei`

DEI

Definition at line 942 of file vtss\_l2\_api.h.

#### 7.289.2.4 tagged

`vtss_vcap_bit_t vtss_vce_tag_t::tagged`

Tagged/untagged frame

Definition at line 943 of file `vtss_l2_api.h`.

#### 7.289.2.5 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`

## 7.290 vtss\_vcl\_port\_conf\_t Struct Reference

VCL port configuration.

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

### Data Fields

- `BOOL dmac_dip`

#### 7.290.1 Detailed Description

VCL port configuration.

Definition at line 878 of file `vtss_l2_api.h`.

#### 7.290.2 Field Documentation

### 7.290.2.1 dmac\_dip

`BOOL vtss_vcl_port_conf_t::dmac_dip`

Enable DMAC/DIP matching (default SMAC/SIP)

Definition at line 879 of file `vtss_I2_api.h`.

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

- `vtss_api/include/vtss_I2_api.h`

## 7.291 vtss\_vid\_mac\_t Struct Reference

MAC Address in specific VLAN.

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

### Data Fields

- `vtss_vid_t vid`
- `vtss_mac_t mac`

### 7.291.1 Detailed Description

MAC Address in specific VLAN.

Definition at line 652 of file `types.h`.

### 7.291.2 Field Documentation

#### 7.291.2.1 vid

`vtss_vid_t vtss_vid_mac_t::vid`

VLAN ID

Definition at line 654 of file `types.h`.

### 7.291.2.2 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](#)

## 7.292 vtss\_vlan\_conf\_t Struct Reference

VLAN configuration.

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

### Data Fields

- [vtss\\_etype\\_t s\\_etype](#)

### 7.292.1 Detailed Description

VLAN configuration.

Definition at line 625 of file vtss\_l2\_api.h.

### 7.292.2 Field Documentation

#### 7.292.2.1 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](#)

## 7.293 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`

### 7.293.1 Detailed Description

VLAN port configuration.

Definition at line 664 of file vtss\_l2\_api.h.

### 7.293.2 Field Documentation

#### 7.293.2.1 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.

#### 7.293.2.2 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.

### 7.293.2.3 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`.

### 7.293.2.4 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`.

### 7.293.2.5 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`

## 7.294 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`

### 7.294.1 Detailed Description

VLAN tag with "arbitrary" TPID.

Definition at line 632 of file `types.h`.

## 7.294.2 Field Documentation

### 7.294.2.1 tpid

`vtss_etype_t` `vtss_vlan_tag_t::tpid`

Tag Protocol Identifier

Definition at line 633 of file types.h.

### 7.294.2.2 pcp

`vtss_tagprio_t` `vtss_vlan_tag_t::pcp`

Priority Code Point

Definition at line 634 of file types.h.

### 7.294.2.3 dei

`BOOL` `vtss_vlan_tag_t::dei`

Drop Eligible Indicator

Definition at line 635 of file types.h.

### 7.294.2.4 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`
-

## 7.295 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`

#### 7.295.1 Detailed Description

VLAN translation group-to-VLAN configuration.

Definition at line 1104 of file `vtss_l2_api.h`.

#### 7.295.2 Field Documentation

##### 7.295.2.1 group\_id

```
u16 vtss_vlan_trans_grp2vlan_conf_t::group_id
```

Group ID

Definition at line 1105 of file `vtss_l2_api.h`.

##### 7.295.2.2 vid

```
vtss_vid_t vtss_vlan_trans_grp2vlan_conf_t::vid
```

VLAN ID

Definition at line 1106 of file `vtss_l2_api.h`.

### 7.295.2.3 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`

## 7.296 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]`

### 7.296.1 Detailed Description

VLAN translation port-to-group configuration.

Definition at line 1098 of file `vtss_l2_api.h`.

### 7.296.2 Field Documentation

#### 7.296.2.1 group\_id

`u16 vtss_vlan_trans_port2grp_conf_t::group_id`

Group ID

Definition at line 1099 of file `vtss_l2_api.h`.

### 7.296.2.2 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](#)

## 7.297 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](#)

### 7.297.1 Detailed Description

VLAN ID configuration.

Definition at line 740 of file vtss\_l2\_api.h.

### 7.297.2 Field Documentation

#### 7.297.2.1 learning

```
BOOL vtss_vlan_vid_conf_t::learning
```

Enable/disable learning

Definition at line 742 of file vtss\_l2\_api.h.

### 7.297.2.2 mirror

`BOOL vtss_vlan_vid_conf_t::mirror`

Enable/disable mirroring

Definition at line 743 of file `vtss_l2_api.h`.

### 7.297.2.3 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`

## 7.298 `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 cmeft_disable`

### 7.298.1 Detailed Description

VStaX configuration for switch.

Definition at line 2324 of file `vtss_l2_api.h`.

### 7.298.2 Field Documentation

### 7.298.2.1 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`.

### 7.298.2.2 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`.

### 7.298.2.3 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`.

### 7.298.2.4 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`.

### 7.298.2.5 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`

## 7.299 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`

#### 7.299.1 Detailed Description

GLAG info.

Definition at line 1611 of file vtss\_l2\_api.h.

#### 7.299.2 Field Documentation

##### 7.299.2.1 upsid

```
vtss_vstax_upsid_t vtss_vstax_glag_entry_t::upsid
```

UPS identifier

Definition at line 1613 of file vtss\_l2\_api.h.

##### 7.299.2.2 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](#)

## 7.300 vtss\_vstax\_port\_conf\_t Struct Reference

VStaX setup for port.

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

## Data Fields

- `u8 ttl`
- `BOOL mirror`

### 7.300.1 Detailed Description

VStaX setup for port.

Definition at line 2358 of file `vtss_l2_api.h`.

### 7.300.2 Field Documentation

#### 7.300.2.1 ttl

`u8 vtss_vstax_port_conf_t::ttl`

TTL, 0-31

Definition at line 2359 of file `vtss_l2_api.h`.

#### 7.300.2.2 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`

## 7.301 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`

### 7.301.1 Detailed Description

UPSID Route Entry.

Definition at line 2425 of file vtss\_l2\_api.h.

### 7.301.2 Field Documentation

#### 7.301.2.1 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.

#### 7.301.2.2 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](#)

## 7.302 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 \[VTSS\\_VSTAX\\_UPSIDS\]](#)

### 7.302.1 Detailed Description

UPSID Route Table.

Definition at line 2431 of file vtss\_l2\_api.h.

### 7.302.2 Field Documentation

#### 7.302.2.1 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.

#### 7.302.2.2 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

## 7.303 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`

### 7.303.1 Detailed Description

VStaX frame header used for reception.

Definition at line 190 of file vtss\_packet\_api.h.

### 7.303.2 Field Documentation

#### 7.303.2.1 valid

`BOOL vtss_vstax_rx_header_t::valid`

TRUE if frame was VStaX tagged

Definition at line 192 of file vtss\_packet\_api.h.

#### 7.303.2.2 sp

`BOOL vtss_vstax_rx_header_t::sp`

TRUE if received on super-prio

Definition at line 193 of file vtss\_packet\_api.h.

#### 7.303.2.3 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.

#### 7.303.2.4 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.

### 7.303.2.5 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`.

### 7.303.2.6 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`

## 7.304 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`

### 7.304.1 Detailed Description

VStaX frame header used for transmission.

Definition at line 453 of file `vtss_packet_api.h`.

## 7.304.2 Field Documentation

### 7.304.2.1 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.

### 7.304.2.2 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.

### 7.304.2.3 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.

### 7.304.2.4 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.

### 7.304.2.5 tci

`vtss_tci_t vtss_vstax_tx_header_t::tci`

VLAN tag information

Definition at line 459 of file vtss\_packet\_api.h.

#### 7.304.2.6 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.

#### 7.304.2.7 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.

#### 7.304.2.8 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.

#### 7.304.2.9 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.

#### 7.304.2.10 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.

### 7.304.2.11 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](#)

## 7.305 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]`

### 7.305.1 Detailed Description

Structure for Wake-On-LAN MAC Address.

Definition at line 1656 of file vtss\_phy\_api.h.

### 7.305.2 Field Documentation

#### 7.305.2.1 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\_OPT\_VCORE\_III 1
- #define VTSS\_FEATURE\_FDMA
- #define VTSS\_FEATURE\_AFI\_FDMA
- #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\_FDMA\_IRQ\_CONTEXT 0
- #define VTSS\_OPT\_FDMA\_DEBUG 0
- #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\_OPT\_VCORE\_III

```
#define VTSS_OPT_VCORE_III 1
```

Internal VCore-III (MIPS) CPU enabled by default

Definition at line 176 of file options.h.

### 8.2.2.68 VTSS\_FEATURE\_FDMA

```
#define VTSS_FEATURE_FDMA
```

Frame DMA

Definition at line 179 of file options.h.

### 8.2.2.69 VTSS\_FEATURE\_AFI\_FDMA

```
#define VTSS_FEATURE_AFI_FDMA
```

FDMA-based Automatic Frame injector supported on EVC-enabled parts

Definition at line 181 of file options.h.

### 8.2.2.70 VTSS\_FEATURE\_VLAN\_TRANSLATION

```
#define VTSS_FEATURE_VLAN_TRANSLATION
```

VLAN Translation

Definition at line 184 of file options.h.

### 8.2.2.71 VTSS\_FEATURE\_SFLOW

```
#define VTSS_FEATURE_SFLOW
```

Statistical flow sampling

Definition at line 185 of file options.h.

### 8.2.2.72 VTSS\_PHY\_10G\_FIFO\_SYNC

```
#define VTSS_PHY_10G_FIFO_SYNC
```

TSFIFO SYNC For 10G PHY

Definition at line 186 of file options.h.

### 8.2.2.73 VIPER\_B\_FIFO\_RESET

```
#define VIPER_B_FIFO_RESET
```

Viper B 1588 FIFO sync

Definition at line 187 of file options.h.

### 8.2.2.74 VTSS\_FEATURE\_QOS\_POLICER\_DLB

```
#define VTSS_FEATURE_QOS_POLICER_DLB
```

DLB policers

Definition at line 328 of file options.h.

### 8.2.2.75 VTSS\_CHIP CU PHY

```
#define VTSS_CHIP CU PHY
```

Cobber PHY chip

Definition at line 540 of file options.h.

### 8.2.2.76 VTSS\_OPT\_TRACE

```
#define VTSS_OPT_TRACE 1
```

Trace enabled by default

Definition at line 548 of file options.h.

### 8.2.2.77 VTSS\_OPT\_FDMA\_IRQ\_CONTEXT

```
#define VTSS_OPT_FDMA_IRQ_CONTEXT 0
```

Deferred interrupt context by default Use of VTSS\_OPT\_FDMA\_VER is the preferred way to indicate which version of the FDMA API is required Make sure noone uses this one anymore

Definition at line 577 of file options.h.

### 8.2.2.78 VTSS\_OPT\_FDMA\_DEBUG

```
#define VTSS_OPT_FDMA_DEBUG 0
```

FDMA debug disabled by default

Definition at line 599 of file options.h.

### 8.2.2.79 VTSS\_OPT\_VAUI\_EQ\_CTRL

```
#define VTSS_OPT_VAUI_EQ_CTRL 6
```

Default equalization control

Definition at line 605 of file options.h.

**8.2.2.80 VTSS\_PHY\_OPT\_VERIPHY**

```
#define VTSS_PHY_OPT_VERIPHY 1
```

VeriPHY enabled by default

Definition at line 613 of file options.h.

**8.2.2.81 VTSS\_FEATURE\_WARM\_START [1/2]**

```
#define VTSS_FEATURE_WARM_START
```

Warm start

Definition at line 624 of file options.h.

**8.2.2.82 VTSS\_FEATURE\_SYNCE\_10G**

```
#define VTSS_FEATURE_SYNCE_10G
```

SYNCE - L1 synchronization feature for 10G PHYs

Definition at line 621 of file options.h.

**8.2.2.83 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.

**8.2.2.84 VTSS\_FEATURE\_WIS**

```
#define VTSS_FEATURE_WIS
```

WAN interface sublayer functionality

Definition at line 623 of file options.h.

**8.2.2.85 VTSS\_FEATURE\_WARM\_START [2/2]**

```
#define VTSS_FEATURE_WARM_START
```

Warm start

Definition at line 624 of file options.h.

**8.2.2.86 VTSS\_ARCH\_MALIBU**

```
#define VTSS_ARCH_MALIBU
```

Used for Malibu-A PHY

Definition at line 625 of file options.h.

**8.2.2.87 VTSS\_ARCH\_MALIBU\_B**

```
#define VTSS_ARCH_MALIBU_B
```

Used for Malibu-B PHY

Definition at line 626 of file options.h.

**8.2.2.88 VTSS\_ARCH\_VENICE\_C**

```
#define VTSS_ARCH_VENICE_C
```

Used for Venice-C PHY

Definition at line 627 of file options.h.

**8.2.2.89 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\_ACTIPHY\_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 5 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

VTSS_SPEED_FIBER_NOT_SUPPORTED_OR_DISABLED	Fiber not supported/ Fiber port disabled
VTSS_SPEED_FIBER_100FX	100BASE-FX
VTSS_SPEED_FIBER_1000X	1000BASE-X
VTSS_SPEED_FIBER_AUTO	Auto detection
VTSS_SPEED_FIBER_DISABLED	Obsolete - use VTSS_SPEED_FIBER_NOT_SUPPORTED_OR_DISABLED 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/bsdःtypes.h>
```

## Data Structures

- struct `vtss_aneg_t`  
*Auto negotiation struct.*
- struct `vtss_vlan_tag_t`
- struct `vtss_mac_t`

- struct `vtss_vid_mac_t`  
*MAC Address.*
- struct `vtss_packet_rx_port_conf_t`  
*MAC Address in specific VLAN.*
- struct `vtss_ipv6_t`  
*Packet registration per port.*
- struct `vtss_ip_addr_t`  
*IPv6 address/mask.*
- struct `vtss_ipv4_network_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 PRId64 "lld"
- #define PRId64 "ldl"
- #define PRId64 "ldx"
- #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 (0xffffffffe)
- #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**  
*Description: CPU Tx group number.*
  - **typedef u16 vtss\_udp\_tcp\_t**  
*Description: UDP/TCP port number.*
  - **typedef u32 vtss\_ip\_t**  
*IPv4 address/mask.*
  - **typedef vtss\_ip\_t vtss\_ipv4\_t**  
*IPv4 address/mask.*
  - **typedef u32 vtss\_prefix\_size\_t**  
*Prefix size.*
  - **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_MA
= -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_PORTS_ONLY, 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_IPV4_UC = 1, VTSS_ROUTING_ENTRY_TYPE_IPV4_MC = 2, VTSS_ROUTING_ENTRY_TYPE_IPV6_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_MILL))
```

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←<br>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←<br>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←<br>LE_SA    | Could not disable sa                                                        |
| VTSS_RC_ERR_MACSEC_COULD_NOT_DEL_R←<br>X_SA     | Could not delete rx sa                                                      |
| VTSS_RC_ERR_MACSEC_COULD_NOT_DEL_T←<br>X_SA     | Could not delete tx sa                                                      |
| VTSS_RC_ERR_MACSEC_PATTERN_NOT_SET              | Pattern not set                                                             |
| VTSS_RC_ERR_MACSEC_HW_RESOURCE_EX←<br>HUSTED    | HW resources exhausted                                                      |
| VTSS_RC_ERR_MACSEC_SCI_ALREADY_EXISTS           | SCI already exists                                                          |
| VTSS_RC_ERR_MACSEC_SC_RESOURCE_NO←<br>T_FOUND   | Could not find SC resources                                                 |
| VTSS_RC_ERR_MACSEC_RX_AN_ALREADY_I←<br>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←<br>XFORM      | Could not program the xform record                                          |
| VTSS_RC_ERR_MACSEC_COULD_NOT_TOGG←<br>LE_SA     | Could not toggle SA                                                         |
| VTSS_RC_ERR_MACSEC_TX_AN_ALREADY_I←<br>N_USE    | Tx AN is in use                                                             |
| VTSS_RC_ERR_MACSEC_ALL_AVAILABLE_SA←<br>_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←<br>USE      | All CP rules of the specific type are in use                                |
| VTSS_RC_ERR_MACSEC_PATTERN_PRIO_NO←<br>T_VALID  | The pattern priority is not valid                                           |
| VTSS_RC_ERR_MACSEC_BUFFER_TOO_SMALL             | Buffer to small, must be greater than<br>VTSS_MACSEC_FRAME_CAPTURE_SIZE_MAX |
| VTSS_RC_ERR_MACSEC_FRAME_TOO_LONG               | Frame length is supposed to be less than the amount<br>of data in the fifo  |
| VTSS_RC_ERR_MACSEC_FRAME_TRUNCATED              | Frame is Truncated                                                          |
| VTSS_RC_ERR_MACSEC_PHY_POWERED_DO←<br>WN        | Phy is powered down, i.e. the MacSec block is not<br>accessible             |
| VTSS_RC_ERR_MACSEC_PHY_NOT_MACSEC←<br>_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←<br>LE_AN    | Could not disable AN                                                        |
| VTSS_RC_ERR_MACSEC_RULE_OUT_OF_RAN←<br>GE       | Rule id is out of range. Must not be larger than<br>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

|                                                    |                                             |
|----------------------------------------------------|---------------------------------------------|
| <code>VTSS_STORM_POLICER_MODE_PORTS_AND_CPU</code> | Police both CPU and front port destinations |
| <code>VTSS_STORM_POLICER_MODE_PORTS_ONLY</code>    | Police front port destinations only         |
| <code>VTSS_STORM_POLICER_MODE_CPU_ONLY</code>      | 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

|                                       |                                 |
|---------------------------------------|---------------------------------|
| <code>VTSS_POLICER_TYPE_MEF</code>    | MEF bandwidth profile           |
| <code>VTSS_POLICER_TYPE_SINGLE</code> | 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

|                                       |                             |
|---------------------------------------|-----------------------------|
| <code>VTSS_VLAN_FRAME_ALL</code>      | Accept all frames           |
| <code>VTSS_VLAN_FRAME_TAGGED</code>   | Accept tagged frames only   |
| <code>VTSS_VLAN_FRAME_UNTAGGED</code> | 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

|                          |                                           |
|--------------------------|-------------------------------------------|
| VTSS_PACKET_REG_NORMAL   | Global registration configuration is used |
| VTSS_PACKET_REG_FORWARD  | Forward normally                          |
| VTSS_PACKET_REG_DISCARD  | Discard                                   |
| VTSS_PACKET_REG_CPU_COPY | Copy to CPU                               |
| VTSS_PACKET_REG_CPU_ONLY | 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

|              |                |
|--------------|----------------|
| VTSS_VDD_1V0 | 1.0V (default) |
| VTSS_VDD_1V2 | 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

|                   |                                   |
|-------------------|-----------------------------------|
| VTSS_IP_TYPE_NONE | Matches "InetAddressType_unknown" |
| VTSS_IP_TYPE_IPV4 | Matches "InetAddressType_ipv4"    |
| VTSS_IP_TYPE_IPV6 | 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

|                                              |                                                                           |
|----------------------------------------------|---------------------------------------------------------------------------|
| <code>VTSS_HQOS_SCH_MODE_NORMAL</code>       | Normal QoS configuration available for non-service traffic only (default) |
| <code>VTSS_HQOS_SCH_MODE_BASIC</code>        | Basic QoS configuration available for non-service traffic only            |
| <code>VTSS_HQOS_SCH_MODE_HIERARCHICAL</code> | 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_fdma_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>
```

### Data Structures

- struct [tag\\_vtss\\_fdma\\_list](#)  
*Software DCB structure.*
- struct [vtss\\_fdma\\_tx\\_info\\_t](#)  
*FDMA Injection Properties.*
- struct [vtss\\_fdma\\_ch\\_cfg\\_t](#)  
*Channel configuration structure.*
- struct [vtss\\_fdma\\_cfg\\_t](#)  
*FDMA configuration structure.*
- struct [vtss\\_fdma\\_throttle\\_cfg\\_t](#)

### Macros

- #define [VTSS\\_FDMA\\_CH\\_CNT](#) 8
- #define [VTSS\\_PHYS\\_PORT\\_CNT](#) 32
- #define [VTSS\\_FDMA\\_DCB\\_SIZE\\_BYTES](#) 24
- #define [VTSS\\_FDMA\\_HDR\\_SIZE\\_BYTES](#) 52
- #define [VTSS\\_FDMA\\_MAX\\_DATA\\_PER\\_DCB\\_BYTES](#) 16380
- #define [VTSS\\_FDMA\\_MIN\\_FRAME\\_SIZE\\_BYTES](#) 64
- #define [VTSS\\_FDMA\\_MIN\\_DATA\\_PER\\_INJ\\_SOF\\_DCB\\_BYTES](#) ([VTSS\\_FDMA\\_HDR\\_SIZE\\_BYTES](#) + [VTSS\\_VSTAX\\_HDR\\_SIZE](#))
- #define [VTSS\\_FDMA\\_MIN\\_DATA\\_PER\\_XTR\\_SOF\\_DCB\\_BYTES](#) ([VTSS\\_FDMA\\_HDR\\_SIZE\\_BYTES](#) + [VTSS\\_VSTAX\\_HDR\\_SIZE](#))
- #define [VTSS\\_FDMA\\_MIN\\_DATA\\_PER\\_NON\\_SOF\\_DCB\\_BYTES](#) 1
- #define [VTSS\\_FDMA\\_MAX\\_FRAME\\_SIZE\\_BYTES](#) 10000
- #define [VTSS\\_OS\\_DCACHE\\_LINE\\_SIZE\\_BYTES](#) 32
- #define [VTSS\\_OS\\_COMPILER\\_ATTRIBUTE\\_ALIGNED](#)(x) \_\_attribute ((aligned(x)))
- #define [VTSS\\_AFI\\_FPS\\_MAX](#) (1000000)
- #define [VTSS\\_FDMA\\_CCM\\_QUOTIENT\\_MAX](#) 3
- #define [VTSS\\_FDMA\\_CCM\\_FREQ\\_LIST\\_LEN](#) 5
- #define [VTSS\\_FDMA\\_CCM\\_FPS\\_MAX](#) [VTSS\\_AFI\\_FPS\\_MAX](#)

### Typedefs

- typedef i32 [vtss\\_fdma\\_ch\\_t](#)  
*Frame DMA channel number. Channels are numbered in range [0; [VTSS\\_FDMA\\_CH\\_CNT](#)].*
- typedef struct [tag\\_vtss\\_fdma\\_list](#) [vtss\\_fdma\\_list\\_t](#)  
*Software DCB structure.*

## Enumerations

- enum `vtss_fdma_afi_type_t` { `VTSS_FDMA_AFI_TYPE_AUTO` = 0, `VTSS_FDMA_AFI_TYPE_FDMA`, `VTSS_FDMA_AFI_TYPE_SWC` }
 

*Select which AFI to use.*
- enum `vtss_fdma_ch_usage_t` { `VTSS_FDMA_CH_USAGE_UNUSED`, `VTSS_FDMA_CH_USAGE_XTR`, `VTSS_FDMA_CH_USAGE_INJ`, `VTSS_FDMA_CH_USAGE_CCM` }
 

*Channel usage.*
- enum `vtss_fdma_dcb_type_t` { `VTSS_FDMA_DCB_TYPE_XTR`, `VTSS_FDMA_DCB_TYPE_INJ`, `VTSS_FDMA_DCB_TYPE_A` }
 

*DCB type identifying a DCB.*

## Functions

- `vtss_rc vtss_fdma_uninit (const vtss_inst_t inst)`

*Uninitialize FDMA.*
- `vtss_rc vtss_fdma_cfg (const vtss_inst_t inst, const vtss_fdma_cfg_t *const cfg)`

*Configure FDMA.*
- `vtss_rc vtss_fdma_dcb_release (const vtss_inst_t inst, vtss_fdma_list_t *const list)`

*Release DCBs.*
- `vtss_rc vtss_fdma_tx (const vtss_inst_t inst, vtss_fdma_list_t *list, vtss_fdma_tx_info_t *const fdma_info, vtss_packet_tx_info_t *const tx_info)`

*Inject a frame.*
- `vtss_rc vtss_fdma_tx_info_init (const vtss_inst_t inst, vtss_fdma_tx_info_t *const fdma_info)`

*Initialize a `vtss_fdma_tx_info_t` structure.*
- `vtss_rc vtss_fdma_afi_cancel (const vtss_inst_t inst, const u8 *const frm_ptr)`

*Cancel an ongoing AFI transmission.*
- `vtss_rc vtss_fdma_afi_frm_cnt (const vtss_inst_t inst, const u8 *const frm_ptr, u64 *const frm_cnt)`

*Get an interim frame count for a given periodically injected frame.*
- `vtss_rc vtss_fdma_dcb_get (const vtss_inst_t inst, u32 dcb_cnt, vtss_fdma_dcb_type_t dcb_type, vtss_fdma_list_t **list)`

*Get one or more DCBs suitable for frame injection.*
- `vtss_rc vtss_fdma_throttle_cfg_get (const vtss_inst_t inst, vtss_fdma_throttle_cfg_t *const cfg)`

*Get current throttle configuration.*
- `vtss_rc vtss_fdma_throttle_cfg_set (const vtss_inst_t inst, const vtss_fdma_throttle_cfg_t *const cfg)`

*Configure throttling.*
- `vtss_rc vtss_fdma_throttle_tick (const vtss_inst_t inst)`

*Generate throttle tick.*
- `vtss_rc vtss_fdma_stats_clr (const vtss_inst_t inst)`

*Clear FDMA statistics.*
- `vtss_rc vtss_fdma_irq_handler (const vtss_inst_t inst, void *const ctxt)`

*FDMA Interrupt Handler.*

### 8.11.1 Detailed Description

Frame DMA API.

### 8.11.2 Macro Definition Documentation

### 8.11.2.1 VTSS\_FDMA\_CH\_CNT

```
#define VTSS_FDMA_CH_CNT 8
```

Number of DMA Channels.

Definition at line 214 of file vtss\_fdma\_api.h.

### 8.11.2.2 VTSS\_PHYS\_PORT\_CNT

```
#define VTSS_PHYS_PORT_CNT 32
```

Number of Port Modules, excluding CPU Port Module.

Definition at line 215 of file vtss\_fdma\_api.h.

### 8.11.2.3 VTSS\_FDMA\_DCB\_SIZE\_BYTES

```
#define VTSS_FDMA_DCB_SIZE_BYTES 24
```

Number of bytes in a DCB.

Definition at line 216 of file vtss\_fdma\_api.h.

### 8.11.2.4 VTSS\_FDMA\_HDR\_SIZE\_BYTES

```
#define VTSS_FDMA_HDR_SIZE_BYTES 52
```

Max(XTR\_HDR\_SZ, INJ\_HDR\_SZ). Worst-case is XTR: 24 bytes for IFH + 4 bytes for queue number + 24 bytes for Rx super-prio artificial IFH). In FDMA API v3+, also injection requires 52 bytes: 24 bytes for IFH + 4 bytes for VLAN tag + 24 bytes for a signature IFH used for multicast injections on 48-ported

Definition at line 217 of file vtss\_fdma\_api.h.

### 8.11.2.5 VTSS\_FDMA\_MAX\_DATA\_PER\_DCB\_BYTES

```
#define VTSS_FDMA_MAX_DATA_PER_DCB_BYTES 16380
```

For both injection and extraction, the maximum number of bytes that one single DCB's associated data area can refer to. If you need to inject or extract larger frames, use multiple DCBs.

Definition at line 300 of file vtss\_fdma\_api.h.

### 8.11.2.6 VTSS\_FDMA\_MIN\_FRAME\_SIZE\_BYTES

```
#define VTSS_FDMA_MIN_FRAME_SIZE_BYTES 64
```

The minimum allowed frame size (excluding IFH and CMD fields, but including FCS) in bytes. This is defined for legacy reasons. The FDMA will automatically adjust any frame below the minimum ethernet size to the minimum ethernet size before transmission.

Definition at line 303 of file vtss\_fdma\_api.h.

### 8.11.2.7 VTSS\_FDMA\_MIN\_DATA\_PER\_INJ\_SOF\_DCB\_BYTES

```
#define VTSS_FDMA_MIN_DATA_PER_INJ_SOF_DCB_BYTES (VTSS_FDMA_HDR_SIZE_BYTES + VTSS_VSTAX_HDR_SIZE)
```

Defines the minimum size in bytes of an injection SOF-DCB's associated data area.

Definition at line 307 of file vtss\_fdma\_api.h.

### 8.11.2.8 VTSS\_FDMA\_MIN\_DATA\_PER\_XTR\_SOF\_DCB\_BYTES

```
#define VTSS_FDMA_MIN_DATA_PER_XTR_SOF_DCB_BYTES (VTSS_FDMA_HDR_SIZE_BYTES + VTSS_VSTAX_HDR_SIZE)
```

Defines the minimum size in bytes of an extraction SOF-DCB's associated data area. Since every DCB can become a SOF DCB, this value also defines the minimum size that the user must allocate per DCB.

Definition at line 308 of file vtss\_fdma\_api.h.

### 8.11.2.9 VTSS\_FDMA\_MIN\_DATA\_PER\_NON\_SOFC\_DCB\_BYTES

```
#define VTSS_FDMA_MIN_DATA_PER_NON_SOFC_DCB_BYTES 1
```

Defines the minimum size in bytes of an injection/extraction non-SOF- DCB's associated data area.

Definition at line 314 of file vtss\_fdma\_api.h.

### 8.11.2.10 VTSS\_FDMA\_MAX\_FRAME\_SIZE\_BYTES

```
#define VTSS_FDMA_MAX_FRAME_SIZE_BYTES 10000
```

The maximum allowed total frame size (excluding IFH and CMD fields) in bytes.

Definition at line 315 of file vtss\_fdma\_api.h.

### 8.11.2.11 VTSS\_OS\_DCACHE\_LINE\_SIZE\_BYTES

```
#define VTSS_OS_DCACHE_LINE_SIZE_BYTES 32
```

The number of bytes one DCache-line is made up of.

Definition at line 318 of file vtss\_fdma\_api.h.

### 8.11.2.12 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 322 of file vtss\_fdma\_api.h.

### 8.11.2.13 VTSS\_AFI\_FPS\_MAX

```
#define VTSS_AFI_FPS_MAX (1000000)
```

Maximum number of frames to inject per second using FDMA-based AFI

Definition at line 930 of file vtss\_fdma\_api.h.

### 8.11.2.14 VTSS\_FDMA\_CCM\_QUOTIENT\_MAX

```
#define VTSS_FDMA_CCM_QUOTIENT_MAX 3
```

Maximum quotient allowed on a given AFI channel.

Definition at line 1752 of file vtss\_fdma\_api.h.

### 8.11.2.15 VTSS\_FDMA\_CCM\_FREQ\_LIST\_LEN

```
#define VTSS_FDMA_CCM_FREQ_LIST_LEN 5
```

#### **AFI:**

The following defines the maximum number of different frequencies (all multiples of each other) that can run on the same channel.

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

Definition at line 1791 of file vtss\_fdma\_api.h.

### 8.11.2.16 VTSS\_FDMA\_CCM\_FPS\_MAX

```
#define VTSS_FDMA_CCM_FPS_MAX VTSS_AFI_FPS_MAX
```

**AFI:**

The following defines the maximum number of frames per second that can be used in vtss\_fdma\_inj\_props\_t's fps member. There is absolutely no guarantee that the actual H/W architecture can reach this number of frames per second.

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

Definition at line 1808 of file vtss\_fdma\_api.h.

## 8.11.3 Typedef Documentation

### 8.11.3.1 vtss\_fdma\_ch\_t

```
typedef i32 vtss_fdma_ch_t
```

Frame DMA channel number. Channels are numbered in range [0; VTSS\_FDMA\_CH\_CNT[.

## FREQUENTLY ASKED QUESTIONS

**Q:** What CPUs does the FDMA work with?

**A:** The FDMA Driver Kit *must* execute on the embedded processor, since an external CPU doesn't have access to the FDMA silicon.

-----oOo-----

**Q:** Can the FDMA be used in parallel with "manual" frame transmission or reception?

**A:** It is not recommended to use the FDMA together with "manual" frame transmission or reception (manual refers to the fact that the CPU spends clock cycles to read or write every single byte to the relevant registers within the switch core). However, for frame transmission, it is possible to have the FDMA working on one set of front ports and the manual transmission working on another set. For frame reception, it is possible to have the FDMA extract from one set of the extraction queues, and have the manual reception working on another set. The sets must be disjunct.

-----oOo-----

**Q:** Are any of the API functions blocking?

**A:** No, none of the functions are blocking. When an API function needs exclusive access to a global variable, it uses a macro (either [VTSS\\_OS\\_INTERRUPT\\_DISABLE\(\)](#) or [VTSS\\_OS\\_SCHEDULER\\_LOCK\(\)](#), depending on the value of the VTSS\_OPT\_FDMA\_IRQ\_CONTEXT preprocessor symbol) to ask for OS-specific support to globally ensure mutual exclusiveness.

-----oOo-----

**Q:** Who allocates memory?

**A:** The API is built in such a way that it is up to the caller to allocate and free memory. Some may wish to allocate all the buffers needed by the FDMA statically, while others may want to allocate it dynamically. If the FDMA code was to implement code for all possible scenarios, it would not be as flexible as it is as of today.

## TERMS AND ABBREVIATIONS

---

**Extraction:**

This term is used by the FDMA silicon and driver kit and is equivalent to what higher levels of software would call Rx, i.e. packet reception. Extraction is abbreviated XTR.

**Injection:**

This term is used by the FDMA silicon and driver kit and is equivalent to what higher levels of software would call Tx, i.e. packet transmission. Injection is abbreviated INJ.

**Packet Module:**

The software module that implements all the calls to the FDMA API.

**Call context:**

The context that a function must be called in. In a multithreaded environment this could be "thread" or "DSR".

**Re-entrancy:**

Some of the functions are allowed to be called simultaneously from several threads.

**DSR:**

Deferred Service Routine. This is the term that eCos uses for the context that is scheduled by an IRQ handler. The context is assumed to be interruptible only by another hardware interrupt, i.e. the scheduler cannot schedule other DSRs or threads while a DSR is running. In Linux, this is known as the "bottom half". A DSR can never wait for critical sections or semaphores or the like.

**DCB:**

In the FDMA driver context, a DCB is normally a software DCB, i.e. an instance of the `vtss_fdma_list_t` structure. Each software DCB embeds a hardware DCB, which is filled by the FDMA driver code and passed to the FDMA silicon, i.e. the higher levels of software need not be concerned about hardware DCBs, but they need to allocate room for them, which is therefore done in the software DCB. The same software DCB type is used for injection and extraction, but some of the fields' meaning differ for the two. Please refer to the `vtss_fdma_list_t` structure for the exact interpretation and Packet Module requirements for the DCBs.

**SOF DCB:**

Start-Of-Frame DCB. The first (software or hardware) DCB in a list of DCBs making up one frame. The list is at least 1 item long.

**EOF DCB:**

End-Of-Frame DCB. The last (software or hardware) DCB in a list of DCBs making up one frame. The list is at least 1 item long, so the EOF DCB may be identical to the SOF DCB.

**Non-SOF DCBs:**

All but the first DCB in a list of (software or hardware) DCBs making up one frame.

**Extraction Queue:**

The chip contains a number of queues for temporarily storing frames destined for the CPU. These so-called extraction queues are of configurable length and are located in the CPU Port Module. They share memory with the CPU Port Module's injection queue. The splitting of memory between the queues is not handled by the FDMA driver code. Likewise, the assignment of a particular type of frames to an extraction queue (which typically takes place in the chip's analyzer (ANA) is not done by the FDMA driver code. The valid range of extraction queue numbers is [VTSS\_PACKET\_RX\_QUEUE\_START; VTSS\_PACKET\_RX\_QUEUE\_END[.

**DMA Channel:**

A DMA channel is used per flow. For extraction each extraction queue is statically mapped to a DMA channel through the call to `vtss_fdma_xtr_cfg()`. For injection, there is no static mapping between a channel number and a front port number, but a channel must still be assigned for injection. Valid channel numbers are in the range [0; VTSS\_FDMA\_CH\_CNT[. An intrinsic priority exists among the channels, in that higher-numbered channels have higher priority. Thus, if two channels serve two different extraction queues, and both report frames present, then the higher numbered channel (which is not necessarily the higher numbered extraction queue!) will get serviced first.

---

**LAYOUT OF INJECTED AND EXTRACTED FRAMES**

**Injection:**

When injecting a frame, the SOF DCB's data pointer must point to an area of at least VTSS\_FDMA\_INJ\_HDR\_SIZE\_BYTES (VTSS\_OPT\_FDMA\_VER == 1) or VTSS\_FDMA\_HDR\_SIZE\_BYTES (VTSS\_OPT\_FDMA\_VER >= 2), which is reserved by the FDMA. The byte following these initial bytes must therefore be the first byte of the frame's DMAC. If VTSS\_OPT\_FDMA\_VER >= 2, then the same buffers can be used for both injection and extraction, and the amount of data to reserve in the beginning of a frame is given by VTSS\_FDMA\_HDR\_SIZE\_BYTES.

**Extraction:**

When an extracted frame is delivered to the callback function, the first VTSS\_FDMA\_XTR\_HDR\_SIZE\_BYTES (VTSS\_OPT\_FDMA\_VER == 1) of the SOF DCB's data area contain the frame's IFH. For VTSS\_OPT\_FDMA\_VER >= 2, the list->ifh\_ptr points to the IFH, which may or may not (depending on architecture) be the same as the list->data member. The list->frm points to the actual frame data. If the frame contains a stack header, then it'll be stripped from the frame (for architectures that carry this in the payload rather than the IFH) and placed in the IFH prior to callback. The list->frm\_ptr points to the actual frame data.

Definition at line 194 of file vtss\_fdma\_api.h.

### 8.11.3.2 vtss\_fdma\_list\_t

```
typedef struct tag_vtss_fdma_list vtss_fdma_list_t
```

Software DCB structure.

The following structure defines a software DCB. Software DCBs can be linked together to form a list of software DCBs. This is the fundamental data structure used to transfer information between a user-level application and the FDMA driver code.

The structure holds the actual hardware DCB needed by the FDMA silicon, a pointer to the associated data area, and other properties such as frame length and allocation length.

The structure is used for both injection and extraction, but some of its members' meaning change slightly in the two cases. The exact interpretation is shown as comments inside the definition below.

### 8.11.4 Enumeration Type Documentation

#### 8.11.4.1 vtss\_fdma\_afi\_type\_t

```
enum vtss_fdma_afi_type_t
```

Select which AFI to use.

On platforms that only support either an FDMA-based or a switch-core-based AFI, select the type it supports or VTSS\_FDMA\_AFI\_TYPE\_AUTO, which will resort to the one it supports.

On platforms that support both AFI types, selecting VTSS\_FDMA\_AFI\_TYPE\_AUTO will cause the FDMA to look at FDMA-only properties, that is, whether frame counting or sequence numbering is enabled, and if so choose the FDMA-based, otherwise it will choose the switch-core based.

**Enumerator**

|                         |                                                        |
|-------------------------|--------------------------------------------------------|
| VTSS_FDMA_AFI_TYPE_AUTO | The FDMA driver chooses an appropriate frame injector. |
| VTSS_FDMA_AFI_TYPE_FDMA | Tell FDMA driver to use the FDMA-based AFI.            |
| VTSS_FDMA_AFI_TYPE_SWC  | Tell FDMA driver to use the switch-core-based AFI.     |

Definition at line 732 of file vtss\_fdma\_api.h.

**8.11.4.2 vtss\_fdma\_ch\_usage\_t**

enum [vtss\\_fdma\\_ch\\_usage\\_t](#)

Channel usage.

A given FDMA channel can either be used for extraction or injection.

**Enumerator**

|                           |                                                                    |
|---------------------------|--------------------------------------------------------------------|
| VTSS_FDMA_CH_USAGE_UNUSED | The channel is not currently in use.                               |
| VTSS_FDMA_CH_USAGE_XTR    | The channel is used/supposed to be used for frame extraction.      |
| VTSS_FDMA_CH_USAGE_INJ    | The channel is used/supposed to be used for frame injection.       |
| VTSS_FDMA_CH_USAGE_CCM    | The channel is used/supposed to be used for period frame injection |

Definition at line 1548 of file vtss\_fdma\_api.h.

**8.11.4.3 vtss\_fdma\_dcb\_type\_t**

enum [vtss\\_fdma\\_dcb\\_type\\_t](#)

DCB type identifying a DCB.

**Enumerator**

|                        |                                                                                      |
|------------------------|--------------------------------------------------------------------------------------|
| VTSS_FDMA_DCB_TYPE_XTR | The DCB is an extraction DCB. Not needed by application.                             |
| VTSS_FDMA_DCB_TYPE_INJ | The DCB is an injection DCB. Needed in call to <a href="#">vtss_fdma_dcb_get()</a> . |
| VTSS_FDMA_DCB_TYPE_AFI | The DCB is an AFI DCB. Needed in call to <a href="#">vtss_fdma_dcb_get()</a> .       |

Definition at line 2581 of file vtss\_fdma\_api.h.

**8.11.5 Function Documentation**

### 8.11.5.1 vtss\_fdma\_uninit()

```
vtss_rc vtss_fdma_uninit (
    const vtss_inst_t inst )
```

Uninitialize FDMA.

Rarely used. Use only if you want to make a controlled shut-down of the FDMA. Disables all FDMA channels and interrupts, and clears pending interrupts.

- Call context:  
Thread
- Re-entrant:  
No

#### Parameters

|             |                                  |
|-------------|----------------------------------|
| <i>inst</i> | [IN]: Target instance reference. |
|-------------|----------------------------------|

#### Returns

VTSS\_RC\_OK on success.  
VTSS\_RC\_ERROR if a supplied parameter was erroneous.

### 8.11.5.2 vtss\_fdma\_cfg()

```
vtss_rc vtss_fdma_cfg (
    const vtss_inst_t inst,
    const vtss_fdma_cfg_t *const cfg )
```

Configure FDMA.

Call this function before any other FDMA API function.

- Call context:  
Thread
- Re-entrant:  
No

#### Parameters

|             |                                  |
|-------------|----------------------------------|
| <i>inst</i> | [IN]: Target instance reference. |
| <i>cfg</i>  | [IN]: FDMA configuration.        |

**Returns**

VTSS\_RC\_OK on success.  
 VTSS\_RC\_ERROR if a supplied parameter was erroneous.

**8.11.5.3 vtss\_fdma\_dcb\_release()**

```
vtss_rc vtss_fdma_dcb_release (
    const vtss_inst_t inst,
    vtss_fdma_list_t *const list )
```

Release DCBs.

This function returns DCBs (injection, extraction, or AFI) back to the FDMA driver.

This is useful in these situations: Extraction: If frame data memory is completely managed by the FDMA driver (`rx_alloc_cb() == NULL`), then the application may choose to pass the frame as is to higher levels of software, i.e. with zero-copy. Once it is handled, the DCBs must be returned to the FDMA driver with a call to this function. If frame data memory is managed by the application (`rx_alloc_cb() != NULL`), then the application should return the list of DCBs it was invoked with in the `rx_cb()` callback handler (with the `alloc_ptr` set to `NULL` if the frame itself is passed to higher levels of software).

Injection (both normal and AFI): This is rarely useful, and the only situation where it makes sense to return injection DCBs to the FDMA driver is when it has allocated the DCBs (using [vtss\\_fdma\\_dcb\\_get\(\)](#)) and then decides not to use them in a call to `vtss_fdma_inj()` afterwards.

- Call context:  
Thread
- Re-entrant:  
Yes

**Parameters**

|             |                                                           |
|-------------|-----------------------------------------------------------|
| <i>inst</i> | [IN]: Target instance reference.                          |
| <i>list</i> | [IN]: The list of DCBs to be returned to the application. |

**Returns**

VTSS\_RC\_OK on success.  
 VTSS\_RC\_ERROR if a supplied parameter was erroneous.

**8.11.5.4 vtss\_fdma\_tx()**

```
vtss_rc vtss_fdma_tx (
    const vtss_inst_t inst,
    vtss_fdma_list_t * list,
```

```
vtss_fdma_tx_info_t *const fdma_info,
vtss_packet_tx_info_t *const tx_info )
```

Inject a frame.

This function takes a NULL-terminated list of software DCBs making up one frame and injects it using the tx info properties given by props.

The DCBs must be obtained using the [vtss\\_fdma\\_dcb\\_get\(\)](#) call.

Once the frame is injected, the configured tx\_done\_cb() function will be called. Once this function returns, the DCBs that the callback function was invoked with are no longer available to the application.

Upon invocation of tx\_done\_cb(), the application can read additional data from the frame's SOF DCB. The fields filled in by the FDMA driver are sw\_tstamp, afi\_frm\_cnt, and afi\_seq\_number.

Upon tx\_done\_cb() the FDMA driver may have modified the actual frame data, and thereby the frm\_ptr that was set during the call to [vtss\\_fdma\\_inj\(\)](#).

- Call context:  
Thread
- Re-entrant:  
Yes

#### Parameters

|                  |                                                                                                                                                                                                                                                                                                                              |
|------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>inst</i>      | [IN]: Target instance reference.                                                                                                                                                                                                                                                                                             |
| <i>list</i>      | [IN]: The list of software DCBs making up the frame. Only the frm_ptr and act_len members must be filled in. The user member is useful for additional data required by the application to identify the frame that is being injected upon the tx_done_cb() callback.                                                          |
| <i>fdma_info</i> | [IN]: Info specifically for use by the FDMA driver. The structure may be allocated on the stack, since <a href="#">vtss_fdma_inj()</a> doesn't keep a pointer to it after the call returns. Initialize with a call to <a href="#">vtss_fdma_tx_info_init()</a> prior to filling it in.                                       |
| <i>tx_info</i>   | [IN]: Pointer to a structure that contains properties on how to inject the frame. Must be initialized with a call to <a href="#">vtss_packet_tx_info_init()</a> before assigning it. The structure may be allocated on the stack, since <a href="#">vtss_fdma_inj()</a> doesn't keep a pointer to it after the call returns. |

#### Returns

VTSS\_RC\_OK on success.

VTSS\_RC\_ERROR on any kind of error. In this case, the DCBs are already returned to the FDMA driver.

#### 8.11.5.5 [vtss\\_fdma\\_tx\\_info\\_init\(\)](#)

```
vtss_rc vtss_fdma_tx_info_init (
    const vtss_inst_t inst,
    vtss_fdma_tx_info_t *const fdma_info )
```

Initialize a [vtss\\_fdma\\_tx\\_info\\_t](#) structure.

**Parameters**

|                  |                               |
|------------------|-------------------------------|
| <i>inst</i>      | [IN]: Target instance         |
| <i>fdma_info</i> | [IN]: Structure to initialize |

**Returns**

VTSS\_RC\_OK unless fdma\_info == NULL.

**8.11.5.6 vtss\_fdma\_afi\_cancel()**

```
vtss_rc vtss_fdma_afi_cancel (
    const vtss_inst_t inst,
    const u8 *const frm_ptr )
```

Cancel an ongoing AFI transmission.

The frame is identified by a pointer to the frame previously used in list->frm\_ptr in the call to [vtss\\_fdma\\_tx\(\)](#).

Once the cancellation has occurred, the afi\_tx\_done callback will be called.

**Parameters**

|                |                       |
|----------------|-----------------------|
| <i>inst</i>    | [IN]: Target instance |
| <i>frm_ptr</i> | [IN]: Frame pointer.  |

**Returns**

VTSS\_RC\_ERROR if no such frame was found. VTSS\_RC\_OK if cancelled successfully.

**8.11.5.7 vtss\_fdma\_afi\_frm\_cnt()**

```
vtss_rc vtss_fdma_afi_frm_cnt (
    const vtss_inst_t inst,
    const u8 *const frm_ptr,
    u64 *const frm_cnt )
```

Get an interim frame count for a given periodically injected frame.

The frame is identified by a pointer to the frame previously used in list->frm\_ptr in the call to [vtss\\_fdma\\_tx\(\)](#). To get a count != 0, counting must be enabled with fdma\_info.afi\_enable\_counting.

**Parameters**

|                |                       |
|----------------|-----------------------|
| <i>inst</i>    | [IN]: Target instance |
| <i>frm_ptr</i> | [IN]: Frame pointer.  |
| <i>frm_cnt</i> | [OUT]: Frame counter. |

**Returns**

VTSS\_RC\_ERROR if no such frame was found. VTSS\_RC\_OK if successful.

**8.11.5.8 vtss\_fdma\_dcb\_get()**

```
vtss_rc vtss_fdma_dcb_get (
    const vtss_inst_t inst,
    u32 dcb_cnt,
    vtss_fdma_dcb_type_t dcb_type,
    vtss_fdma_list_t ** list )
```

Get one or more DCBs suitable for frame injection.

The FDMA API is the owner/maintainer of all DCBs. In order to be able to inject a frame, the application must request DCBs from the FDMA driver and fill in appropriate fields (see description under `vtss_fdma_inj()`) prior to calling `vtss_fdma_inj()`.

One or more DCBs may be requested in one go. If the application regrets that it has requested the DCBs, it may call `vtss_fdma_dcbs_release()` to hand them back to the FDMA driver.

If more than one DCB is requested, the FDMA driver will hook them together with the DCBs' next field. The last DCB's next field will be set to NULL by the FDMA driver.

Special DCBs are required for AFI frames, so in addition to the number of DCBs, also a `dcb_type` parameter must be provided. The `dcb_type` must only be set to `VTSS_FDMA_DCB_TYPE_INJ` or `VTSS_FDMA_DCB_TYPE_AFI`.

If the requested number of DCBs are not available of the specified type, the function returns `VTSS_RC_INCOMPLETE`. It is up to the application to implement the logic that can facilitate waiting for DCBs, if it is required that a given frame must be transmitted. It could, e.g. wait for a semaphore in the thread that calls `vtss_fdma_inj()` and signal that semaphore whenever the `tx_done_cb()` gets invoked.

- Call context:  
Thread
- Re-entrant:  
Yes

**Parameters**

|                 |                                                      |
|-----------------|------------------------------------------------------|
| <i>inst</i>     | [IN]: Target instance reference.                     |
| <i>dcb_cnt</i>  | [IN]: Number of DCBs.                                |
| <i>dcb_type</i> | [IN]: DCB type requested.                            |
| <i>list</i>     | [OUT]: Pointer receiving a pointer to the first DCB. |

**Returns**

VTSS\_RC\_OK on success.  
 VTSS\_RC\_ERROR if a supplied parameter was erroneous.  
 VTSS\_RC\_INCOMPLETE if `dcb_cnt` DCBs aren't available.

### 8.11.5.9 `vtss_fdma_throttle_cfg_get()`

```
vtss_rc vtss_fdma_throttle_cfg_get (
    const vtss_inst_t inst,
    vtss_fdma_throttle_cfg_t *const cfg )
```

Get current throttle configuration.

Returns the current throttling configuration.

- Call context:  
Any
- Re-entrant:  
Yes, but doesn't really make sense.

#### Parameters

|             |                                                                                 |
|-------------|---------------------------------------------------------------------------------|
| <i>inst</i> | [IN]: Target instance reference.                                                |
| <i>cfg</i>  | [OUT]: Pointer to structure that receives the current throttling configuration. |

#### Returns

VTSS\_RC\_OK on success, VTSS\_RC\_ERROR if channel indicated by ch is not configured for extraction.

### 8.11.5.10 `vtss_fdma_throttle_cfg_set()`

```
vtss_rc vtss_fdma_throttle_cfg_set (
    const vtss_inst_t inst,
    const vtss_fdma_throttle_cfg_t *const cfg )
```

Configure throttling.

Configure throttling. See [vtss\\_fdma\\_throttle\\_cfg\\_t](#) for a description of the throttle feature and the use of this function.

- Call context:  
Any
- Re-entrant:  
Yes, but doesn't really make sense.

#### Parameters

|             |                                     |
|-------------|-------------------------------------|
| <i>inst</i> | [IN]: Target instance reference.    |
| <i>cfg</i>  | [IN]: New throttling configuration. |

**Returns**

VTSS\_RC\_OK on success, VTSS\_RC\_ERROR if channel indicated by ch is not configured for extraction.

**8.11.5.11 vtss\_fdma\_throttle\_tick()**

```
vtss_rc vtss_fdma_throttle_tick (
    const vtss_inst_t inst )
```

Generate throttle tick.

See [vtss\\_fdma\\_throttle\\_cfg\\_t](#) for a description of the throttle feature and the use of this function.

- Call context:  
Any
- Re-entrant:  
Yes, but doesn't really make sense to call this function from more than one thread.

**Parameters**

|             |                                  |
|-------------|----------------------------------|
| <i>inst</i> | [IN]: Target instance reference. |
|-------------|----------------------------------|

**Returns**

VTSS\_RC\_OK - always.

**8.11.5.12 vtss\_fdma\_stats\_clr()**

```
vtss_rc vtss_fdma_stats_clr (
    const vtss_inst_t inst )
```

Clear FDMA statistics.

- Call context:  
Thread
- Re-entrant:  
Yes

**Parameters**

|             |                                  |
|-------------|----------------------------------|
| <i>inst</i> | [IN]: Target instance reference. |
|-------------|----------------------------------|

**Returns**

VTSS\_RC\_OK on success.  
 VTSS\_RC\_ERROR if supplied parameter was erroneous.

**8.11.5.13 vtss\_fdma\_irq\_handler()**

```
vtss_rc vtss_fdma_irq_handler (
    const vtss_inst_t inst,
    void *const ctxt )
```

**FDMA Interrupt Handler.**

This function is the heart-beat of all FDMA silicon communication. The driver code enables interrupts on the FDMA silicon and expects this function to be called whenever the FDMA generates interrupts. The function is not re-entrant, and it must *not* be interrupted by other non-interrupt code. This function does not call any of the [VTSS\\_OS\\_INTERRUPT\\_DISABLE\(\)](#) or [VTSS\\_OS\\_SCHEDULER\\_LOCK\(\)](#) macros.

The function first checks for extraction channels interrupting and calls back the relevant xtr\_cb() functions with one frame at a time. The function calls back for every frame that the FDMA has extracted since last call. Then it checks for injection completion, takes care of retransmission if that is needed, and calls back the relevant inj\_post\_cb() functions for every frame that has been injected.

This means that both xtr\_cb() and inj\_post\_cb() callback functions are called from the same context as [vtss\\_fdma\\_irq\\_handler\(\)](#) is called. BEWARE!!

To better understand what is required by the caller of this function, let's branch on the two basic options:

- **Interrupt Driven Operation:**  
 If interrupts are supported, it is important that the FDMA interrupt is disabled by the "top half" or ISR if it finds out that the interrupt is for the FDMA. If the OS supports top and bottom halves like Linux (a.k.a. I $\leftrightarrow$ SRs and DSRs in eCos), it is recommended to call [vtss\\_fdma\\_irq\\_handler\(\)](#) in the bottom half/DSR rather than in the top half/ISR. In the bottom half/DSR case, it is ensured that no other bottom halves/DSRs or threads can preempt the handler. Once the handler returns, the caller should clear and re-enable top-level FDMA interrupts. In this scenario VTSS\_OPT\_FDMA\_IRQ\_CONTEXT should be defined to 0, causing the thread code (all other vtss\_fdma\_XXX() functions) to use the [VTSS\\_OS\\_SCHEDULER\\_LOCK\(\)](#)/[UNLOCK\(\)](#) functions.

If the OS doesn't support top/bottom halves, but still supports interrupt handling, [vtss\\_fdma\\_irq\\_handler\(\)](#) may be called directly from the ISR, which should still start by disabling top-level FDMA interrupts, then call the handler, and finally clear and re-enable them. In this scenario VTSS\_OPT\_FDMA\_IRQ\_CONTEXT be defined to 1. This causes the thread code (all other vtss\_fdma\_XXX() functions) to use the [VTSS\\_OS\\_INTERRUPT\\_DISABLE\(\)](#)/[RESTORE\(\)](#) functions whenever it is using or updating state members also used by the interrupt handler.

- **Polled Operation:**  
 Since the FDMA driver code reads registers directly in the FDMA silicon to figure out which channels are interrupting, and since it allows for "no channels want the CPU's attention", it is possible to use a polled approach rather than an interrupt driven approach. In a single-threaded application you can call it once in the round-robin trip, because it cannot be interrupted by any other code. In a multi-threaded application you will have to disable the scheduler before calling it, and re-enable it afterwards. This ensures that no other functions can call e.g. vtss\_fdma\_inj() while servicing the "interrupt". It is not good enough to disable interrupts, because the code calls macros like "output this debug message to the console", which may end up in the OS kernel, which in turn may schedule the [vtss\\_fdma\\_irq\\_handler\(\)](#) function out in favor of another thread.

- **Call context:**  
 OS-Specific. See description below.
- **Re-entrant:**  
**NO!!!**

**Parameters**

|              |                                                                                                                        |
|--------------|------------------------------------------------------------------------------------------------------------------------|
| <i>inst</i>  | [IN]: Target instance reference.                                                                                       |
| <i>cntxt</i> | [IN]: A user-defined argument, which is passed to the inj_post_cb() and xtr_cb() callback functions.<br>Rarely needed. |

**Returns**

VTSS\_RC\_OK on success.  
VTSS\_RC\_ERROR if @inst parameter was erroneous.

## 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`  
`= 0x7424, 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 0xFFFFFFF2
```

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<br>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 |
|----------------|------------------|

**Parameters**

|                    |                                                                                                        |
|--------------------|--------------------------------------------------------------------------------------------------------|
| <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. |
|-------------|----------------------------|

**Parameters**

|                |                                                                                                                                                      |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| <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           |
| 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,
    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\_MAX\_CNT - 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**  
*VStA 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_TYPEETYPE`, `VTSS_VCE_TYPE_LLIC`, `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)

- `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.*
- `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)
  - Delete MAC address entries learned on (UPSID, UPSPN).*
- `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)  
*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: $\cdot\cdot$ /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. |

**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 <a href="#">vtss_vlan_trans_grp2vlan_conf_t</a> . 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 <a href="#">vtss_vlan_trans_port2grp_conf_t</a> . |

**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 <a href="#">vtss_vlan_trans_port2grp_conf_t</a> . |
| <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\_I3\_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\_I3\_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\_I3\_api.h.

## 8.17.3 Enumeration Type Documentation

### 8.17.3.1 vtss\_I3\_rleg\_common\_mode\_t

```
enum vtss_I3_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\_I3\_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)`

- `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.*

- **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 SGPIO 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 SGPIO 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 scheduler 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, othewise 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     |
| VTSS_GPIO_ALT_2  |                          |

Definition at line 567 of file vtss\_misc\_api.h.

#### 8.19.4.7 vtss\_sgpio\_mode\_t

enum `vtss_sgpio_mode_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\_sgpio\_bmode\_t

enum `vtss_sgpio_bmode_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

|                    |                                  |
|--------------------|----------------------------------|
| <code>group</code> | [IN] Trace group                 |
| <code>conf</code>  | [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<br>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_ecos.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_C TZ`(val32) <your impl>
- #define `VTSS_OS_C TZ64`(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](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\_M ALLOC

```
#define VTSS_OS_M ALLOC(
    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\_F REE

```
#define VTSS_OS_F REE(
    ptr,
    flags ) <your impl>
```

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 101 of file vtss\_os\_custom.h.

### 8.24.2.15 VTSS\_OS\_R AND

```
#define VTSS_OS_R AND( ) <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](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

|                  |                                              |
|------------------|----------------------------------------------|
| <code>val</code> | [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

|                    |                                                     |
|--------------------|-----------------------------------------------------|
| <code>size</code>  | [IN] Number of bytes to allocate.                   |
| <code>flags</code> | [IN] See <code>vtss_mem_flags_t</code> 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

|              |                                                                                       |
|--------------|---------------------------------------------------------------------------------------|
| <i>ptr</i>   | [IN] Pointer previously obtained with call to <a href="#">vtss_callout_malloc()</a> . |
| <i>flags</i> | [IN] See <a href="#">vtss_mem_flags_t</a> 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](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\_MALLOC

```
#define VTSS_OS_MALLOC( \
    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\_FREE

```
#define VTSS_OS_FREE( \
    ptr, \
    flags ) free(ptr)
```

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 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_TWOSTEP     | 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<br>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><b>Can be set by:</b></p> <p>Luton26: Y Jaguar1: Y Serval: Y<br/>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><b>Can be set by:</b></p> <p>Luton26: Y Jaguar1: Y Serval: Y<br/>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><b>Can be set by:</b></p> <p>Luton26: Y Jaguar1: Y Serval: Y<br/>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.<br>Generated by Doxygen |
| <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`
  - 10G PHY prbs monitor Configuration
- struct `vtss_phy_10g_ib_storage_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 prbs monitor Configuration
- struct `vtss_phy_10g_pcs_prbs_gen_conf_t`
  - 10G PHY Packet generator configuration
- struct `vtss_phy_10g_pcs_prbs_mon_conf_t`
  - 10G PHY Packet Monitor configuration
- struct `vtss_phy_10g_prbs_gen_conf_t`
  - 10G PHY prbs gen 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_led_conf_t`
  - LED Mode selection.
- struct `vtss_gpio_10g_gpio_mode_t`
  - 10G Phy GPIO mode

- struct [vtss\\_phy\\_10g\\_fw\\_status\\_t](#)  
*Firmware status.*
- struct [vtss\\_phy\\_10g\\_i2c\\_slave\\_conf\\_t](#)  
*10G Phy I2C Master Interface for SFP Module Configuration*

## 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\_REL\_THRESH\_EV 0x40000000
- #define VTSS\_PHY\_EWIS\_RELIP\_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\_TX\_CHAR\_ENC\_CNT\_THRESH\_EV 0x00000020
- #define VTSS\_PHY\_10G\_RX\_BLK\_DEC\_CNT\_THRESH\_EV 0x00000040
- #define VTSS\_PHY\_10G\_TX\_BLK\_ENC\_CNT\_THRESH\_EV 0x00000080
- #define VTSS\_PHY\_10G\_RX\_SEQ\_CNT\_THRESH\_EV 0x00000100
- #define VTSS\_PHY\_10G\_TX\_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\_SYNC\_MODE,  
 VTSS\_PHY\_WAN\_SYNC\_MODE, VTSS\_PHY\_LAN\_MIXED\_SYNC\_MODE, VTSS\_PHY\_WAN\_MIXED\_SYNC\_MODE,  
 VTSS\_PHY\_REPEAT\_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_SR2_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_APP_AUTO`, `VTSS_IB_APP_MANUAL`, `VTSS_IB_APP_FREEZE`, `VTSS_IB_APP_RESET`,  
`VTSS_IB_APP_RESTART`, `VTSS_IB_APP_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_recvrd_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\_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_MAS,
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\_INTO,
 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\_ID,
 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\_LEGR\_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.*
- enum `vtss_gpio_10g_led_mode_t` {
 `VTSS_10G_GPIO_LED_NONE` = 0, `VTSS_10G_GPIO_LED_TX_LINK_STATUS` = 1, `VTSS_10G_GPIO_LED_TX_LINK_TX`  
 = 3, `VTSS_10G_GPIO_LED_TX_LINK_RX_DATA` = 4,  
`VTSS_10G_GPIO_LED_RX_LINK_STATUS` = 5, `VTSS_10G_GPIO_LED_RX_LINK_RX_DATA` = 7,  
`VTSS_10G_GPIO_LED_RX_LINK_TX_RX_DATA` = 8 }
   
*LED Modes.*
- enum `vtss_gpio_10g_led_blink_t` { `VTSS_10G_GPIO_LED_BLINK_50MS`, `VTSS_10G_GPIO_LED_BLINK_100MS`,  
`VTSS_10G_GPIO_LED_BLINK_NONE` }
   
*LED Blink Interval.*

## 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 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_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.*
  - `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_st *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_reset` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no)

*i2c reset*

  - `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_i2c_slave_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_10g_i2c_slave_conf_t` \*const i2c\_conf)
- Sets the configuration for the I2C slave.*
- `vtss_rc vtss_phy_10g_i2c_slave_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_10g_i2c_slave_conf_t` \*i2c\_conf)
- Gets the I2C configuration parameters.*
- `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 2531 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 2533 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 2598 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 2638 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 2639 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 2601 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 2602 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 2603 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 2604 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 2606 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 2607 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 2608 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 2609 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 2610 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 2611 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 2612 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 2613 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 2614 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 2615 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 2616 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 2617 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 2618 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 2619 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 2620 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 2621 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 2622 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 2623 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 2624 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 2625 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 2626 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 2627 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 2628 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 2629 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 2630 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 2637 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 RXLOL interrupt - only on 8488

PHY RX LOL interrupt - 8256 specific

Definition at line 2638 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 2639 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 2641 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 2642 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 2643 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 2644 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 2645 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 2646 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 2647 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 2648 of file vtss\_phy\_10g\_api.h.

**8.30.2.60 VTSS\_PHY\_10G\_HIGHBER\_EV**

```
#define VTSS_PHY_10G_HIGHBER_EV 0x00001000
```

PHY high bit Error - 8256 specific

Definition at line 2649 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 2651 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 2651 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 2652 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 2653 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 2654 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 2655 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 2656 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 2657 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 2660 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 2661 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 2662 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 2663 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 2632 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 2666 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:<br>848X: Dual clock (XREFCK=156,25 MHz, WREFCK=155,52 MHz), no recovered clock output<br>Venice: Single clock (XREFCK), no recovered clock output                                                                                                                                                                                                                                                                            |
| VTSS_PHY_1G_MODE              | 8488: 1G pass-through mode<br>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:<br>if hl_clk_synth == 1:<br>8488: Single clock (XREFCK=156,25 MHz), recovered clock output enabled<br>Venice: Single clock (XREFCK=156,25 MHz), recovered clock output enabled<br>if hl_clk_synth == 0:<br>8488: Dual clock (XREFCK=156,25 MHz, SREFCK=156,25 MHz), recovered clock output enabled<br>Venice: Dual clock (XREFCK=156,25 MHz, SREFCK=156,25 MHz), recovered clock output enabled                             |
| VTSS_PHY_WAN_SYNCE_MODE       | WAN SyncE:<br>if hl_clk_synth == 1:<br>8488: Single clock (WREFCK=155,52 MHz or 622,08 MHz), recovered clock output enabled<br>Venice: Single clock (XREFCK=156,25 MHz), recovered clock output enabled<br>if hl_clk_synth == 0:<br>8488: Dual clock (WREFCK=155,52 MHz or 622,08 MHz, SREFCK=155,52 MHz), recovered clock output enabled<br>Venice: Dual clock (XREFCK=156,25 MHz, SREFCK=155,52 MHz), recovered clock output enabled |
| VTSS_PHY_LAN_MIXED_SYNCE_MODE | 8488: Channels are in different modes, channel being configured is in LAN<br>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<br>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 |
| VTSS_PHY_RXAUI_XFI        | RXAUI <-> XFI - Interface mode. Only for Venice  |
| VTSS_PHY_SGMII_LANE_0_XFI | SGMII <-> XFI - LANE 0. Only for Venice          |
| VTSS_PHY_SGMII_LANE_3_XFI | SGMII <-> XFI - LANE 3. Only for Venice          |
| VTSS_PHY_SFI_XFI          | 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**

|                      |                                      |
|----------------------|--------------------------------------|
| VTSS_RECVRD_RXCLKOUT | RXCLKOUT is used for recovered clock |
| VTSS_RECVRD_TXCLKOUT | 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**

|                           |                        |
|---------------------------|------------------------|
| VTSS_RECVRDCLK_CDR_DIV_64 | recovered clock is /64 |
| VTSS_RECVRDCLK_CDR_DIV_66 | 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_APP_IB_SFP_PLUS_ZR | SFP+ ZR module.        |
| VTSS_APP_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 Id_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<br>(direct connection)         |
| VTSS_6G_LINK_LONG_RANGE  | distance between 6G macro and serdes macro of link partner is more<br>(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

|                                                    |                                                          |
|----------------------------------------------------|----------------------------------------------------------|
| <code>VTSS_RECVRD_CLKOUT_DISABLE</code>            | recovered clock output is disabled                       |
| <code>VTSS_RECVRD_CLKOUT_LINE_SIDE_RX_CLOCK</code> | recovered clock output is derived from Lineside Rx clock |
| <code>VTSS_RECVRD_CLKOUT_LINE_SIDE_TX_CLOCK</code> | 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

|                                           |                              |
|-------------------------------------------|------------------------------|
| <code>VTSS_PHY_10G_SREFCLK_156_25</code>  | 156,25 MHz                   |
| <code>VTSS_PHY_10G_SREFCLK_125_00</code>  | 125,00 MHz                   |
| <code>VTSS_PHY_10G_SREFCLK_155_52</code>  | 155,52 MHz                   |
| <code>VTSS_PHY_10G_SREFCLK_INVALID</code> | 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

|                                           |                                           |
|-------------------------------------------|-------------------------------------------|
| <code>VTSS_PHY_10G_CLK_FULL_RATE</code>   | LAN:332.25 MHz, WAN:311.04MHz, 1G:125MHz  |
| <code>VTSS_PHY_10G_CLK_DIVIDE_BY_2</code> | LAN:161.12 MHz, WAN:155.52MHz, 1G:62.5MHz |
| <code>VTSS_PHY_10G_CLK_INVALID</code>     | 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  |
| 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    |
| 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 | 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 loopback                                                           |
| 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 |

## Enumerator

|                     |                                                                                        |
|---------------------|----------------------------------------------------------------------------------------|
| 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

|                            |                                     |
|----------------------------|-------------------------------------|
| VTSS_PHY_10G_POWER_ENABLE  | Enable Phy power for all sublayers  |
| VTSS_PHY_10G_POWER_DISABLE | 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

|                                       |                                            |
|---------------------------------------|--------------------------------------------|
| VTSS_PHY_10G_PMA_TO_FROM_XAUI_NORMAL  | PMA_0/1 to XAUI_0/1. 8487: XAUI 0 to PMA 0 |
| VTSS_PHY_10G_PMA_TO_FROM_XAUI_CROSSED | PMA_0/1 to XAUI_1/0. 8487: XAUI 1 to PMA 0 |

**Enumerator**

|                                              |                                                 |
|----------------------------------------------|-------------------------------------------------|
| VTSS_PHY_10G_PMA_0_TO_FROM_XAUI_0_T↔O_XAUI_1 | PMA 0 to/from XAUI 0 and to XAUI 1              |
| VTSS_PHY_10G_PMA_0_TO_FROM_XAUI_1_T↔O_XAUI_0 | PMA 0 to/from XAUI 1 and to XAUI 0              |
| VTSS_PHY_10G_PMA_1_TO_FROM_XAUI_0_T↔O_XAUI_1 | PMA 1 to/from XAUI 0 and to XAUI 1. VSC8487:N/A |
| VTSS_PHY_10G_PMA_1_TO_FROM_XAUI_1_T↔O_XAUI_0 | 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 |
|-----------------------------------|----------------------------------------------------------------------------------------------------------------|

## Enumerator

|                                   |                                                                                                                                                                         |
|-----------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 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_RXFAULT     | 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                                                                                                                                       |
| 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 ,for MALIBU per-channel GPIO also to be selected                                                                                                            |
| 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           |
| 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_I↔_B_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_IN↔_TR    | 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            |
| VTSS_10G_GPIO_INTR_SGNL_HDATA_ACT_RX         | 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                     |

## Enumerator

|                                                              |                                                     |
|--------------------------------------------------------------|-----------------------------------------------------|
| VTSS_10G_GPIO_INTR_SGNL_MACSEC_1588_↔<br>SFD_LANE            | Ethernet channel disable                            |
| VTSS_10G_GPIO_INTR_SGNL_LINE_S_TX_FAULT                      | MACSEC,1588 SFD lane                                |
| VTSS_10G_GPIO_INTR_SGNL_LPCS1G_LATEN_↔<br>CY0_OR_EWIS_BIT0   | TX fault                                            |
| VTSS_10G_GPIO_INTR_SGNL_LPCS1G_LATEN_↔<br>CY1_OR_EWIS_BIT1   | LPCS1G latency 0 in case of 1G mode or EWIS BIT 0   |
| VTSS_10G_GPIO_INTR_SGNL_LPCS1G_CHAR_↔<br>_POS0_OR_EWIS_BIT2  | LPCS1G latency 0 in case of 1G mode or EWIS BIT 1   |
| VTSS_10G_GPIO_INTR_SGNL_LPCS1G_CHAR_↔<br>_POS1_OR_EWIS_WORD0 | LPCS1G Char pos 0 in case of 1G mode or EWIS BIT 2  |
| VTSS_10G_GPIO_INTR_SGNL_LPCS1G_CHAR_↔<br>_POS2_OR_EWIS_WORD1 | LPCS1G Char pos 1 in case of 1G mode or EWIS word 0 |
| VTSS_10G_GPIO_INTR_SGNL_LPCS1G_CHAR_↔<br>_POS3_OR_EWIS_WORD2 | LPCS1G Char pos 2 in case of 1G mode or EWIS word 1 |
| VTSS_10G_GPIO_INTR_SGNL_MACSEC_IGR_↔<br>PRED_VAR0            | Macsec ingress predictor var 0                      |
| VTSS_10G_GPIO_INTR_SGNL_MACSEC_IGR_↔<br>PRED_VAR1            | Macsec ingress predictor var 1                      |
| VTSS_10G_GPIO_INTR_SGNL_KR_ACTV_2GPIO                        | KR activity                                         |
| VTSS_10G_GPIO_INTR_SGNL_DFT_TX_2GPIO                         | DFT transmit                                        |
| VTSS_10G_GPIO_INTR_SGNL_RESERVED                             | Reserved for future use                             |
| VTSS_10G_GPIO_INTR_SGNL_EXE_LST_2GPIO_↔<br>O_0               | EXE LST to GPIO 0                                   |
| VTSS_10G_GPIO_INTR_SGNL_EXE_LST_2GPIO_↔<br>O_1               | EXE LST to GPIO 1                                   |
| VTSS_10G_GPIO_INTR_SGNL_EXE_LST_2GPIO_↔<br>O_2               | EXE LST to GPIO 2                                   |
| VTSS_10G_GPIO_INTR_SGNL_EXE_LST_2GPIO_↔<br>O_3               | EXE LST to GPIO 3                                   |
| VTSS_10G_GPIO_INTR_SGNL_EXE_LST_2GPIO_↔<br>O_4               | EXE LST to GPIO 4                                   |
| VTSS_10G_GPIO_INTR_SGNL_LINK_HCD_2GPIO_↔<br>O_0              | Link HCD to GPIO 0                                  |
| VTSS_10G_GPIO_INTR_SGNL_LINK_HCD_2GPIO_↔<br>O_1              | Link HCD to GPIO 1                                  |
| VTSS_10G_GPIO_INTR_SGNL_LINK_HCD_2GPIO_↔<br>O_2              | Link HCD to GPIO 2                                  |
| VTSS_10G_GPIO_INTR_SGNL_ETH_1G_ENA                           | Ethernet 1G enable                                  |
| VTSS_10G_GPIO_INTR_SGNL_H_KR_8b10b_2↔<br>GPIO                | KR 8b10b to GPIO                                    |
| VTSS_10G_GPIO_INTR_SGNL_H_KR_10Gb_2↔<br>PIO                  | 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          |
| VTSS_10G_GPIO_INTRPT_LIGR_FIFO  | FC Buffer interrupt         |
| VTSS_10G_GPIO_INTRPT_LEGRI_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 |

**Enumerator**

|                                             |                         |
|---------------------------------------------|-------------------------|
| 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.

**8.30.4.40 vtss\_gpio\_10g\_led\_mode\_t**

enum [vtss\\_gpio\\_10g\\_led\\_mode\\_t](#)

LED Modes.

**GPIO input modes****Enumerator**

|                                      |                                                     |
|--------------------------------------|-----------------------------------------------------|
| VTSS_10G_GPIO_LED_TX_LINK_STATUS     | Disable LED configuration on given GPIO             |
| VTSS_10G_GPIO_LED_TX_LINK_TX_DATA    | Display LINE Tx link status                         |
| VTSS_10G_GPIO_LED_TX_LINK_RX_DATA    | Display LINE Tx link status and Rx data activity    |
| VTSS_10G_GPIO_LED_RX_LINK_STATUS     | Display LINE Rx link status and Tx,Rx data activity |
| VTSS_10G_GPIO_LED_RX_LINK_RX_DATA    | Display LINE Rx link status                         |
| VTSS_10G_GPIO_LED_RX_LINK_TX_RX_DATA | Display LINE Rx link status and Rx data activity    |

Definition at line 2480 of file vtss\_phy\_10g\_api.h.

#### 8.30.4.41 vtss\_gpio\_10g\_led\_blink\_t

enum [vtss\\_gpio\\_10g\\_led\\_blink\\_t](#)

LED Blink Interval.

GPIO LED modes

Enumerator

|                                               |                             |
|-----------------------------------------------|-----------------------------|
| <a href="#">VTSS_10G_GPIO_LED_BLINK_100MS</a> | Led Blink interval is 50 ms |
| <a href="#">VTSS_10G_GPIO_LED_BLINK_NONE</a>  | Led Blink interval is 100ms |

Definition at line 2494 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. |

**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> ←<br><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 og 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.                             |

**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\_reset()

```
vtss_rc vtss_phy_10g_i2c_reset (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no )
```

i2c reset

#### Parameters

|                |                                 |
|----------------|---------------------------------|
| <i>inst</i>    | [IN] Target instance reference. |
| <i>port_no</i> | [IN] Port number                |

#### Returns

Return code.

### 8.30.5.93 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.

### 8.30.5.94 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.

**8.30.5.95 vtss\_phy\_10g\_i2c\_slave\_conf\_set()**

```
vtss_rc vtss_phy_10g_i2c_slave_conf_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_10g_i2c_slave_conf_t *const i2c_conf )
```

Sets the configuration for the I2C slave.

**Parameters**

|                 |                                   |
|-----------------|-----------------------------------|
| <i>inst</i>     | [IN] Target instance reference    |
| <i>port_no</i>  | [IN] Port number                  |
| <i>i2c_conf</i> | [IN] Sets configuration for Slave |

**Returns**

Return code.

**8.30.5.96 vtss\_phy\_10g\_i2c\_slave\_conf\_get()**

```
vtss_rc vtss_phy_10g_i2c_slave_conf_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_10g_i2c_slave_conf_t * i2c_conf )
```

Gets the I2C configuration parameters.

**Parameters**

|                 |                                    |
|-----------------|------------------------------------|
| <i>inst</i>     | [IN] Target instance reference.    |
| <i>port_no</i>  | [IN] Port number                   |
| <i>i2c_conf</i> | [OUT] Gets configuration for Slave |

**Returns**

Return code.

**8.30.5.97 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\\_cntl\\_t](#)  
*PHY MAC SerDes PCS Control, Reg16E3.*
- struct [vtss\\_phy\\_media\\_serdes\\_pcs\\_cntl\\_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, LINK1000_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, LINK1000BASE_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_FI_1000BX,`  
`VTSS_PHY_MEDIA_IF_FI_100FX,`  
`VTSS_PHY_MEDIA_IF_AMS_CU_PASSTHRU, VTSS_PHY_MEDIA_IF_AMS_FI_PASSTHRU, 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 pArtner'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` \*mcb\_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\_reddump\_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\_level 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.*

*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)*

- `vtss_rc vtss_phy_sd6g_mac_serdes_conf` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no)

*Function for Configuring MAC i/f 6G serdes (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

|                                    |                                                                                                                                                                                                       |
|------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>LINK1000_ACTIVITY</code>     | 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.           |
| <code>LINK100_ACTIVITY</code>      | No link in 1000BASE-T/Valid 1000BASE-T link. Blink or pulse-stretch = Valid 1000BASE-T link with activity present                                                                                     |
| <code>LINK10_ACTIVITY</code>       | No link in 100BASE-T/Valid 100BASE-T link. Blink or pulse-stretch = Valid 100BASE-T link with activity present                                                                                        |
| <code>LINK100_1000_ACTIVITY</code> | No link in 10BASE-T/Valid 10BASE-T link. Blink or pulse-stretch = Valid 10BASE-T link with activity present                                                                                           |
| <code>LINK10_1000_ACTIVITY</code>  | 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. |

## Enumerator

|                                      |                                                                                                                                                                                                    |
|--------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 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        |
| 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.                                                                                                                                                      |
| BASE100_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             |

**Enumerator**

|                                   |                                                                                        |
|-----------------------------------|----------------------------------------------------------------------------------------|
| 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

|                                              |                               |
|----------------------------------------------|-------------------------------|
| <code>VTSS_PHY_FAST_LINK_FAIL_ENABLE</code>  | Enable fast link failure pin  |
| <code>VTSS_PHY_FAST_LINK_FAIL_DISABLE</code> | 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

|                                              |                                                                                        |
|----------------------------------------------|----------------------------------------------------------------------------------------|
| VTSS_PHY_MAC_SERD_PCS_SGMII_IN_PRE_↔<br>NONE | MAC SerDes PCS Control, SGMII Input Preamble for<br>100BaseX - No Preamble Req'd       |
| VTSS_PHY_MAC_SERD_PCS_SGMII_IN_PRE_↔<br>ONE  | MAC SerDes PCS Control, SGMII Input Preamble for<br>100BaseX - One-Byte Preamble Req'd |
| VTSS_PHY_MAC_SERD_PCS_SGMII_IN_PRE_↔<br>TWO  | MAC SerDes PCS Control, SGMII Input Preamble for<br>100BaseX - Two-Byte Preamble Req'd |
| VTSS_PHY_MAC_SERD_PCS_SGMII_IN_PRE_↔<br>RSVD | MAC SerDes PCS Control, SGMII Input Preamble for<br>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_↔<br>NO_ERROR   | Media SerDes PCS Control, Most Recent Clause 37<br>ANEG Exchg - Table 37-3 |
| VTSS_PHY_MEDIA_SERD_PCS_Rem_Fault_↔<br>OFFLINE    | Media SerDes PCS Control, Most Recent Clause 37<br>ANEG Exchg              |
| VTSS_PHY_MEDIA_SERD_PCS_Rem_Fault_L↔<br>INK_FAIL  | Media SerDes PCS Control, Most Recent Clause 37<br>ANEG Exchg              |
| VTSS_PHY_MEDIA_SERD_PCS_Rem_Fault_A↔<br>NEG_ERROR | Media SerDes PCS Control, Most Recent Clause 37<br>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_↔<br>NORMAL | Force AMS Override to Force Selection - Normal          |
| VTSS_PHY_MEDIA_FORCE_AMS_SELECTION_↔<br>SERDES | Force AMS Override to Force Selection - SerDes<br>Media |
| VTSS_PHY_MEDIA_FORCE_AMS_SELECTION_↔<br>COPPER | Force AMS Override to Force Selection - Copper<br>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

|                     |                                                        |
|---------------------|--------------------------------------------------------|
| VTSS_PHY_FREQ_25M   | 25 MHz                                                 |
| VTSS_PHY_FREQ_125M  | 125 MHz                                                |
| VTSS_PHY_FREQ_3125M | 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

|                           |                                                                                                                                                                                                                                           |
|---------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| VTSS_PHY_CLK_SQUELCH_MAX  | 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). |
| VTSS_PHY_CLK_SQUELCH_MED  | Same as VTSS_PHY_CLK_SQUELCH_MAX 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.                      |
| VTSS_PHY_CLK_SQUELCH_MIN  | Squelch only when the link is not up                                                                                                                                                                                                      |
| VTSS_PHY_CLK_SQUELCH_NONE | 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←<br>_4 | PasswdLen=4 bytes |
| VTSS_WOL_PASSWD_LEN←<br>_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

|                                |                                                  |
|--------------------------------|--------------------------------------------------|
| VTSS_100FX_FEFI_NORMAL         | Normal FEFI Operation, as specified by Reg23E3.1 |
| VTSS_100FX_FEFI_FORCE_SUPPRESS | Force FEFI, as specified by Reg23E3.0            |
| VTSS_100FX_FEFI_FORCE_ENABLE   | 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**

|                |                                                         |
|----------------|---------------------------------------------------------|
| <i>inst</i>    | [IN] Target instance reference.                         |
| <i>port_no</i> | [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 pArtners'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 bits 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                                                                                                                                                                                                                                                                              |
| <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.<br>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.                                                           |
| <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. |
|-------------|---------------------------------|

**Parameters**

|                 |                                                                                                                                                         |
|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| <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 not 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:<br>bits 11:8: Port Address (0-3) mode: bits 7:4: Loopback Type:<br>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.

### 8.31.5.73 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<br>mode: bits 15:12: 0x8= Fibre Media or 0x9=SGMII/QSGMII MAC mode: bits 11:8: Port Address (0-3)<br>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 <a href="#">vtss_phy_wol_conf_t</a> 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 <a href="#">vtss_phy_wol_conf_t</a> 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.  |

**Parameters**

|                 |                                                                                                                                                                |
|-----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>mcb_bus</i>  | [INOUT] If set to 2 the returned data is for LCPLL/RComp else the <i>mcb_bus</i> 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 <a href="#">vtss_rcpll_status_t</a> 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 <a href="#">vtss_rcpll_status_t</a> 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 <a href="#">vtss_lcpll_status_t</a> 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, Customizable.  |
| <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, Customizeable 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 = $20 * \log_{10}(\text{mse\_dbl})$ ; 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 $\text{mse\_dbl} = \text{mse} / (1024 * 2048)$ ; Note: Convert to Double and Scale by $2^{21} = 1024 * 2048$ $\text{mse\_dbl} = 20 * \log_{10}(\text{mse\_dbl})$ ; Note: Convert to dB Nominal Computed Values for $\text{mse\_dbl}$ 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.

## 8.31.5.99 vtss\_phy\_sd6g\_mac\_serdes\_conf()

```
vtss_rc vtss_phy_sd6g_mac_serdes_conf (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no )
```

Function for Configuring MAC i/f 6G serdes (applicable only on VIPER,ELISE)

## Parameters

|                |                                 |
|----------------|---------------------------------|
| <i>inst</i>    | [IN] Target instance reference. |
| <i>port_no</i> | [IN] Any phy port with the chip |

## 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 \*cctxt, 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_ETH_IP_PTP`, `VTSS_PHY_TS_ENCAP_ETH_MPLS_IP_PTP`, `VTSS_PHY↓`  
`TS_ENCAP_ETH_MPLS_ETH_PTP`, `VTSS_PHY_TS_ENCAP_ETH_MPLS_ETH_IP_PTP`,  
`VTSS_PHY_TS_ENCAP_ETH_MPLS_ACH_PTP`, `VTSS_PHY_TS_ENCAP_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_↓`  
`ONE` }
 

*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↓`  
`2DM` }
 

*Y.1731 OAM delay measurement method.*
- enum `vtss_phy_ts_ietaf_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_ietaf_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_CL↓`  
`IENT`, `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_OUT_PTP` }

*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)

*Get the alternative external 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)

- `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 \*cntxt)
 

*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 \*\*cntxt)
 

*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` 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 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 ingress 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)

- `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)
  - Enable N-Phase sampler.*
- `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)
  - N-Phase sampler status get.*
  - 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 *ctxt, 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 ctxt 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 encapsulation i.e. OAM-over-ETH

#### Enumerator

|                                            |                                                  |
|--------------------------------------------|--------------------------------------------------|
| <code>VTSS_PHY_TS_PTP_ENGINE_ID_0</code>   | PTP engine 0                                     |
| <code>VTSS_PHY_TS_PTP_ENGINE_ID_1</code>   | PTP engine 1                                     |
| <code>VTSS_PHY_TS_OAM_ENGINE_ID_2A</code>  | OAM engine 2A, no PTP support                    |
| <code>VTSS_PHY_TS_OAM_ENGINE_ID_2B</code>  | OAM engine 2B, no PTP; only OAM-over-ETH support |
| <code>VTSS_PHY_TS_ENGINE_ID_INVALID</code> | 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_DELAYM_DMM | Two-way delay measurement method                |
| VTSS_PHY_TS_IETF_MPLS_ACH_OAM_DELAYM_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

|                                                             |                        |
|-------------------------------------------------------------|------------------------|
| <a href="#">VTSS_PHY_TS_IETF_MPLS_ACH_OAM_TS_FORMAT_PTP</a> | 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

|                                     |                       |
|-------------------------------------|-----------------------|
| <a href="#">VTSS_PHY_TS_4_BYTE</a>  | Nano second timestamp |
| <a href="#">VTSS_PHY_TS_10_BYTE</a> | 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

|                                               |            |
|-----------------------------------------------|------------|
| <a href="#">VTSS_PHY_TS_CLOCK_FREQ_125M</a>   | 125 MHz    |
| <a href="#">VTSS_PHY_TS_CLOCK_FREQ_15625M</a> | 156.25 MHz |
| <a href="#">VTSS_PHY_TS_CLOCK_FREQ_200M</a>   | 200 MHz    |
| <a href="#">VTSS_PHY_TS_CLOCK_FREQ_250M</a>   | 250 MHz    |
| <a href="#">VTSS_PHY_TS_CLOCK_FREQ_500M</a>   | 500 MHz    |
| <a href="#">VTSS_PHY_TS_CLOCK_FREQ_MAX</a>    | 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

|                                              |                                                                                                        |
|----------------------------------------------|--------------------------------------------------------------------------------------------------------|
| <code>VTSS_PHY_TS_CLOCK_SRC_EXTERNAL</code>  | External source                                                                                        |
| <code>VTSS_PHY_TS_CLOCK_SRC_CLIENT_RX</code> | 10G: XAUI lane 0 recovered clock, 1G: MAC RX clock (note: direction is opposite to 10G, i.e. PHY->MAC) |
| <code>VTSS_PHY_TS_CLOCK_SRC_CLIENT_TX</code> | 10G: XAUI lane 0 recovered clock, 1G: MAC TX clock (note: direction is opposite to 10G, i.e. MAC->PHY) |
| <code>VTSS_PHY_TS_CLOCK_SRC_LINE_RX</code>   | Received line clock                                                                                    |
| <code>VTSS_PHY_TS_CLOCK_SRC_LINE_TX</code>   | transmitted line clock                                                                                 |
| <code>VTSS_PHY_TS_CLOCK_SRC_INTERNAL</code>  | 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 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

|                                                  |                                |
|--------------------------------------------------|--------------------------------|
| <code>VTSS_PHY_TS_RX_TIMESTAMP_POS_IN_PTP</code> | 4 reserved bytes in PTP header |
| <code>VTSS_PHY_TS_RX_TIMESTAMP_POS_AT_END</code> | 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

|                                       |                                                                                                                 |
|---------------------------------------|-----------------------------------------------------------------------------------------------------------------|
| <code>VTSS_PHY_TS_TC_OP_MODE_A</code> | Sub local time at ingress and add at egress from CF                                                             |
| <code>VTSS_PHY_TS_TC_OP_MODE_B</code> | RX_timestamp using reserved bytes or append at the end as defined in <code>vtss_phy_ts_rxtimestamp_pos_t</code> |
| <code>VTSS_PHY_TS_TC_OP_MODE_C</code> | 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

|                                         |                                 |
|-----------------------------------------|---------------------------------|
| <code>VTSS_PHY_TS_NPHASE_PPS_O</code>   | N-Phase sampler for PPS_O       |
| <code>VTSS_PHY_TS_NPHASE_PPS_RI</code>  | N-Phase sampler for PPS_RI      |
| <code>VTSS_PHY_TS_NPHASE_EGR_SOF</code> | N-Phase sampler for egress SOF  |
| <code>VTSS_PHY_TS_NPHASE_ING_SOF</code> | N-Phase sampler for ingress SOF |
| <code>VTSS_PHY_TS_NPHASE_LS</code>      | N-Phase sampler for Load/Save   |
| <code>VTSS_PHY_TS_NPHASE_MAX</code>     | 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

|                      |                                |
|----------------------|--------------------------------|
| <code>inst</code>    | [IN] handle to an API instance |
| <code>port_no</code> | [IN] port number               |
| <code>saved</code>   | [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        |

**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>encap_type</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_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.

#### 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_type</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 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.

**8.32.5.47 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\\_MIM\\_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)) }

- 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

|                                      |                               |
|--------------------------------------|-------------------------------|
| <code>VTSS_PORT_MAX_TAGS_NONE</code> | No extra tags allowed         |
| <code>VTSS_PORT_MAX_TAGS_ONE</code>  | Single tag allowed            |
| <code>VTSS_PORT_MAX_TAGS_TWO</code>  | 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

|                                      |                    |
|--------------------------------------|--------------------|
| <code>VTSS_PORT_LOOP_DISABLE</code>  | No port loop       |
| <code>VTSS_PORT_LOOP_PCS_HOST</code> | 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

|                                         |                                                 |
|-----------------------------------------|-------------------------------------------------|
| <code>VTSS_PORT_FORWARD_ENABLED</code>  | Forward in both directions                      |
| <code>VTSS_PORT_FORWARD_DISABLED</code> | Forwarding and learning disabled                |
| <code>VTSS_PORT_FORWARD_INGRESS</code>  | Forward frames from port only                   |
| <code>VTSS_PORT_FORWARD_EGRESS</code>   | 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](#)

- 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\_TYPE\_ETYPE.*
- struct [vtss\\_qce\\_frame\\_llc\\_t](#)  
*Frame data for VTSS\_QCE\_TYPE\_LLCC.*
- struct [vtss\\_qce\\_frame\\_snap\\_t](#)  
*Frame data for VTSS\_QCE\_TYPE\_SNAP.*
- struct [vtss\\_qce\\_frame\\_ipv4\\_t](#)  
*Frame data for VTSS\_QCE\_TYPE\_IPV4.*
- struct [vtss\\_qce\\_frame\\_ipv6\\_t](#)  
*Frame data for VTSS\_QCE\_TYPE\_IPV6.*
- 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_dscp_emode_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_LLC`, `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

|                                      |                                                                                 |
|--------------------------------------|---------------------------------------------------------------------------------|
| <code>VTSS_DSCP_EMODE_DISABLE</code> | DSCP not remarked                                                               |
| <code>VTSS_DSCP_EMODE_REMARK</code>  | DSCP remarked with DSCP value from analyzer                                     |
| <code>VTSS_DSCP_EMODE_REMAP</code>   | 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

|                                  |                |
|----------------------------------|----------------|
| <code>VTSS_QCE_TYPE_ANY</code>   | Any frame type |
| <code>VTSS_QCE_TYPE_ETYPE</code> | Ethernet Type  |
| <code>VTSS_QCE_TYPE_LLC</code>   | LLC            |
| <code>VTSS_QCE_TYPE_SNAP</code>  | SNAP           |
| <code>VTSS_QCE_TYPE_IPV4</code>  | IPv4           |
| <code>VTSS_QCE_TYPE_IPV6</code>  | 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\_LLCC, 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  |
| 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.<br>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\_SYNCE\_CLK\_A 0
- #define VTSS\_SYNCE\_CLK\_B 1
- #define VTSS\_SYNCE\_CLK\_MAX 2

## Typedefs

- typedef u32 vtss\_sync\_clk\_port\_t  
*Identification of a output clock port.*

## Enumerations

- enum vtss\_sync Divider\_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_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 <a href="#">vtss_sync_clock_out_t</a> 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_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_SYNCE_CLK_A or VTSS_SYNCE_CLK_B)   |
| <i>conf</i>     | [IN] pointer to a <a href="#">vtss_sync_clock_out_t</a> 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_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_SYNCE_CLK_A or VTSS_SYNCE_CLK_B)  |
| <i>conf</i>     | [IN] pointer to a <a href="#">vtss_sync_clock_in_t</a> 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_clock_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_CLOCK_A or VTSS_SYNC_CLOCK_B) |
| <i>conf</i>     | [IN] pointer to a <a href="#">vtss_sync_clock_in_t</a> 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_E`←
   
`XT_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_RE`←
   
`SERVED_LEN_32BIT, TS_INTERNAL_FMT_SUB_ADD_LEN_44BIT_CF62,`
  
`TS_INTERNAL_FMT_RESERVED_LEN_48BIT_CF_3_0, TS_INTERNAL_FMT_RESERVED_LEN_48BI`←
   
`T_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\\_ewisaisl\\_cons\\_act\\_s](#)  
*eWIS AIS-L consequent actions*
- struct [vtss\\_ewisrdil\\_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`  
*Trail 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 eWIS overhead passthru configuration.*
- `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.*
- `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.*
- `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.*
- `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_txti_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_ewis_tti_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.*
- `vtss_rc vtss_ewis_section_txti_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_ewis_tti_t` \*const txti)
 

*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_txti_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_ewis_tti_t` \*const txti)
 

*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_txti_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_ewis_tti_t` \*const txti)
 

*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 WIS 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
```

Unequipped 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 Unequipped 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                   |
| VTSS_WIS_TEST_MODE_MAX             | 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

|                        |                                                                                  |
|------------------------|----------------------------------------------------------------------------------|
| EWIS_PRBS31_SINGLE_ERR | Inject a single bit error (=> error counter incrementing by 3                    |
| EWIS_PRBS31_SAT_ERR    | 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**

|                |                                      |
|----------------|--------------------------------------|
| <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.

**8.42.5.2 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\_ttxi\_set()**

```
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.

**Parameters**

|                |                                  |
|----------------|----------------------------------|
| <i>inst</i>    | [IN] Target instance reference.  |
| <i>port_no</i> | [IN] Port number.                |
| <i>ttxi</i>    | [IN] pointer to transmitted tti. |

**Returns**

Return code.

**8.42.5.18 vtss\_ewis\_section\_ttxi\_get()**

```
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.

**Parameters**

|                |                                   |
|----------------|-----------------------------------|
| <i>inst</i>    | [IN] Target instance reference.   |
| <i>port_no</i> | [IN] Port number.                 |
| <i>ttxi</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\_txti\_set()

```
vtss_rc vtss_ewis_path_txti_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_ewis_txti_t *const txti )
```

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>txti</i>    | [IN] pointer to transmitted tti. |

##### Returns

Return code.

#### 8.42.5.21 vtss\_ewis\_path\_txti\_get()

```
vtss_rc vtss_ewis_path_txti_get (
    const vtss_inst_t inst,
```

```
const vtss_port_no_t port_no,  
vtss_ewis_tti_t *const txti )
```

Get the configured Path 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.

**8.42.5.22 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**

XAU API.

```
#include <vtss/api/types.h>
```

**8.43.1 Detailed Description**

XAU 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, 238

AMPLITUDE\_POINTS  
    vtss\_phy\_10g\_api.h, 848

ach\_opt  
    vtss\_phy\_ts\_oam\_engine\_flow\_conf\_t, 403  
    vtss\_phy\_ts\_ptp\_engine\_flow\_conf\_t, 410

acknowledge  
    vtss\_phy\_10g\_clause\_37\_adv\_t, 251  
    vtss\_port\_clause\_37\_adv\_t, 427

acl\_hit  
    vtss\_packet\_rx\_info\_t, 206

acl\_idx  
    vtss\_packet\_rx\_info\_t, 206

act\_len  
    tag\_vtss\_fdma\_list, 37

action  
    vtss\_ace\_t, 56  
    vtss\_acl\_port\_conf\_t, 64  
    vtss\_ece\_t, 87  
    vtss\_phy\_ts\_engine\_action\_t, 368  
    vtss\_qce\_t, 472  
    vtss\_vce\_t, 534

action\_gen  
    vtss\_phy\_ts\_engine\_action\_t, 368

action\_ptp  
    vtss\_phy\_ts\_engine\_action\_t, 368

active  
    vtss\_irq\_status\_t, 172  
    vtss\_phy\_10g\_kr\_status\_aneg\_t, 272  
    vtss\_phy\_10g\_prbs\_mon\_conf\_t, 309

addr  
    vtss\_ip\_addr\_t, 165  
    vtss\_ipv6\_t, 170  
    vtss\_mac\_t, 184  
    vtss\_phy\_ts\_ip\_conf\_t, 393  
    vtss\_wol\_mac\_addr\_t, 555

addr\_match\_mode  
    vtss\_phy\_ts\_eth\_conf\_t, 373

addr\_match\_select  
    vtss\_phy\_ts\_eth\_conf\_t, 373

address  
    vtss\_ip\_network\_t, 166  
    vtss\_ipv4\_network\_t, 167  
    vtss\_ipv6\_network\_t, 169

adv\_10g  
    vtss\_phy\_10g\_base\_kr\_id\_adv\_abil\_t, 242

adv\_1g  
    vtss\_phy\_10g\_base\_kr\_id\_adv\_abil\_t, 242

adv\_dis  
    port\_custom\_conf\_t, 34

advertisement  
    vtss\_phy\_10g\_clause\_37\_control\_t, 253  
    vtss\_port\_clause\_37\_control\_t, 429

afi\_buf\_cnt  
    vtss\_fdma\_cfg\_t, 146

afi\_done\_cb  
    vtss\_fdma\_cfg\_t, 146

afi\_enable\_counting  
    vtss\_fdma\_tx\_info\_t, 154

afi\_enable\_sequence\_numbering  
    vtss\_fdma\_tx\_info\_t, 154

afi\_fps  
    vtss\_fdma\_tx\_info\_t, 153

afi\_frm\_cnt  
    tag\_vtss\_fdma\_list, 39

afi\_seq\_number  
    tag\_vtss\_fdma\_list, 39

afi\_sequence\_number\_offset  
    vtss\_fdma\_tx\_info\_t, 155

afi\_type  
    vtss\_fdma\_tx\_info\_t, 154

agc  
    vtss\_phy\_10g\_ib\_conf\_t, 263

aged  
    vtss\_mac\_table\_entry\_t, 185  
    vtss\_mac\_table\_status\_t, 188

aggr\_code  
    vtss\_packet\_tx\_info\_t, 226

aggr\_intrpt  
    vtss\_gpio\_10g\_gpio\_mode\_t, 157

aggr\_rx\_disable  
    vtss\_packet\_frame\_info\_t, 195  
    vtss\_packet\_port\_info\_t, 197

aggr\_tx\_disable  
    vtss\_packet\_frame\_info\_t, 195  
    vtss\_packet\_port\_info\_t, 198

ais\_on\_lof  
    vtss\_ewis\_aisl\_cons\_act\_s, 100

ais\_on\_los  
    vtss\_ewis\_aisl\_cons\_act\_s, 100

aisl  
    vtss\_ewis\_cons\_act\_s, 105

alloc\_ptr  
    tag\_vtss\_fdma\_list, 38

amp\_range  
    vtss\_phy\_10g\_vscope\_scan\_status\_t, 329

ampl

vtss\_phy\_10g\_base\_kr\_conf\_t, 241  
 an\_enable  
     vtss\_phy\_10g\_base\_kr\_autoneg\_t, 239  
 an\_reset  
     vtss\_phy\_10g\_base\_kr\_autoneg\_t, 239  
 an\_restart  
     vtss\_phy\_10g\_base\_kr\_autoneg\_t, 239  
 ana\_sync  
     vtss\_ewis\_test\_status\_s, 131  
 aneg  
     vtss\_phy\_10g\_base\_kr\_status\_t, 244  
     vtss\_phy\_conf\_t, 335  
     vtss\_port\_status\_t, 459  
 aneg\_complete  
     vtss\_port\_sgmii\_aneg\_t, 457  
     vtss\_port\_status\_t, 459  
 aneg\_enable  
     vtss\_phy\_tbi\_conf\_t, 361  
 aneg\_pd\_detect  
     vtss\_phy\_media\_serd\_pcs\_cntl\_t, 350  
 aneg\_restart  
     vtss\_phy\_mac\_serd\_pcs\_cntl\_t, 347  
 apc\_bit\_mask  
     vtss\_phy\_10g\_ib\_conf\_t, 265  
 apc\_eqz\_offs\_par\_cfg  
     vtss\_phy\_10g\_mode\_t, 287  
 apc\_host\_eqz\_ld\_ctrl  
     vtss\_phy\_10g\_mode\_t, 287  
 apc\_host\_ld\_ctrl  
     vtss\_phy\_10g\_mode\_t, 286  
 apc\_ib\_regulator  
     vtss\_phy\_10g\_mode\_t, 288  
 apc\_ib\_regulator\_t  
     vtss\_phy\_10g\_api.h, 864  
 apc\_line\_eqz\_ld\_ctrl  
     vtss\_phy\_10g\_mode\_t, 287  
 apc\_line\_ld\_ctrl  
     vtss\_phy\_10g\_mode\_t, 286  
 apc\_offs\_ctrl  
     vtss\_phy\_10g\_mode\_t, 285  
 arp  
     vtss\_ace\_frame\_arp\_t, 40  
     vtss\_ace\_t, 58  
 arrived\_tagged  
     vtss\_packet\_rx\_header\_t, 201  
 asymmetric\_pause  
     vtss\_phy\_10g\_clause\_37\_adv\_t, 250  
     vtss\_phy\_aneg\_t, 331  
     vtss\_port\_clause\_37\_adv\_t, 427  
 auto\_clear\_ls  
     vtss\_phy\_ts\_init\_conf\_t, 389  
 automatic  
     vtss\_learn\_mode\_t, 182  
 autoneg  
     port\_custom\_conf\_t, 33  
     vtss\_phy\_10g\_base\_kr\_train\_aneg\_t, 245  
     vtss\_phy\_10g\_clause\_37\_status\_t, 255  
 BOOLEAN\_STORAGE\_COUNT  
     vtss\_phy\_10g\_api.h, 847  
 BOOL  
     types.h, 620  
 bad\_crc  
     vtss\_phy\_10g\_pkt\_mon\_conf\_t, 301  
 base\_address  
     vtss\_l3\_common\_conf\_t, 174  
 base\_port\_no  
     vtss\_phy\_type\_t, 417  
 ber  
     vtss\_phy\_10g\_pkt\_mon\_conf\_t, 302  
     vtss\_phy\_10g\_vscope\_scan\_conf\_t, 328  
 ber\_cnt  
     vtss\_phy\_pcs\_cnt\_t, 352  
 bin  
     vtss\_packet\_tx\_info\_t, 228  
 bist\_mode  
     vtss\_phy\_10g\_prbs\_mon\_conf\_t, 308  
 bit\_count  
     vtss\_sgpio\_conf\_t, 494  
 bit\_errors  
     vtss\_phy\_10g\_ib\_status\_t, 266  
 blink  
     vtss\_gpio\_10g\_led\_conf\_t, 159  
 block\_lock  
     vtss\_phy\_10g\_kr\_status\_aneg\_t, 274  
     vtss\_phy\_10g\_status\_t, 323  
 block\_lock\_latched  
     vtss\_phy\_pcs\_cnt\_t, 352  
 bmode  
     vtss\_sgpio\_conf\_t, 494  
 bottom\_up  
     vtss\_phy\_ts\_mpls\_conf\_t, 397  
 bpdu\_cpu\_only  
     vtss\_packet\_rx\_reg\_t, 221  
 bpdu\_queue  
     vtss\_packet\_rx\_queue\_map\_t, 218  
 bpdu\_reg  
     vtss\_packet\_rx\_port\_conf\_t, 216  
 bridge  
     vtss\_port\_counters\_t, 434  
 broadcast  
     vtss\_policer\_ext\_t, 422  
 bypass\_in\_api  
     vtss\_phy\_10g\_fifo\_sync\_t, 257  
 byte\_limit\_per\_tick  
     vtss\_fdma\_throttle\_cfg\_t, 151  
 bytes  
     vtss\_counter\_pair\_t, 70  
 c  
     vtss\_phy\_10g\_ib\_conf\_t, 263  
 c0  
     vtss\_phy\_10g\_base\_kr\_conf\_t, 240  
 c0\_ob\_tap\_result  
     vtss\_phy\_10g\_kr\_status\_train\_t, 276  
 c1  
     vtss\_phy\_10g\_base\_kr\_conf\_t, 241  
 c\_ctrl

vtss\_phy\_10g\_ob\_status\_t, 293  
c\_intrpt  
    vtss\_gpio\_10g\_gpio\_mode\_t, 157  
CALIB\_DONE  
    vtss\_phy\_ts\_nphase\_status\_t, 400  
CALIB\_ERR  
    vtss\_phy\_ts\_nphase\_status\_t, 399  
CHIP\_PORT\_UNUSED  
    vtss\_port\_api.h, 1077  
cal\_done  
    vtss\_lcpll\_status\_t, 181  
cal\_error  
    vtss\_lcpll\_status\_t, 181  
    vtss\_rcpll\_status\_t, 485  
cal\_not\_done  
    vtss\_rcpll\_status\_t, 485  
cb  
    vtss\_ts\_timestamp\_alloc\_t, 509  
cbs  
    vtss\_dlb\_policer\_conf\_t, 76  
ccm\_quotient\_max  
    vtss\_fdma\_ch\_cfg\_t, 150  
cf  
    vtss\_dlb\_policer\_conf\_t, 75  
cf\_update  
    vtss\_phy\_ts\_ptp\_engine\_action\_t, 408  
cfg  
    vtss\_phy\_conf\_1g\_t, 334  
cfg0  
    vtss\_phy\_10g\_mode\_t, 286  
cfi  
    vtss\_ace\_vlan\_t, 59  
    vtss\_tci\_t, 501  
channel\_conf  
    vtss\_phy\_10g\_init\_parm\_t, 270  
channel\_id  
    vtss\_phy\_10g\_auto\_failover\_conf\_t, 237  
    vtss\_phy\_10g\_id\_t, 269  
    vtss\_phy\_10g\_mode\_t, 283  
    vtss\_phy\_type\_t, 417  
channel\_map  
    vtss\_phy\_ts\_engine\_flow\_conf\_t, 369  
    vtss\_phy\_ts\_generic\_action\_t, 383  
    vtss\_phy\_ts\_oam\_engine\_action\_t, 401  
    vtss\_phy\_ts\_ptp\_engine\_action\_t, 407  
channel\_type  
    vtss\_phy\_ts\_ach\_conf\_t, 364  
chip\_no  
    vtss\_debug\_info\_t, 72  
    vtss\_debug\_lock\_t, 73  
    vtss\_fdma\_ch\_cfg\_t, 150  
    vtss\_packet\_rx\_meta\_t, 212  
    vtss\_port\_map\_t, 446  
chip\_port  
    vtss\_port\_map\_t, 446  
    vtss\_vstax\_tx\_header\_t, 554  
chk\_ing\_modified  
    vtss\_phy\_ts\_init\_conf\_t, 390  
cir  
    vtss\_dlb\_policer\_conf\_t, 76  
ckout\_sel  
    vtss\_phy\_10g\_ckout\_conf\_t, 249  
ckout\_sel\_  
    vtss\_phy\_10g\_api.h, 871  
ckout\_sel\_t  
    vtss\_phy\_10g\_api.h, 861  
clear  
    vtss\_debug\_info\_t, 72  
clk\_freq  
    vtss\_phy\_ts\_init\_conf\_t, 387  
clk\_mode  
    vtss\_phy\_ts\_ptp\_engine\_action\_t, 408  
clk\_mstr\_t  
    vtss\_phy\_10g\_api.h, 865  
clk\_sel\_no  
    vtss\_phy\_10g\_host\_clk\_conf\_t, 260  
    vtss\_phy\_10g\_line\_clk\_conf\_t, 279  
clk\_src  
    vtss\_phy\_ts\_init\_conf\_t, 387  
cm  
    vtss\_dlb\_policer\_conf\_t, 75  
cm1  
    vtss\_phy\_10g\_base\_kr\_conf\_t, 240  
cm\_ob\_tap\_result  
    vtss\_phy\_10g\_kr\_status\_train\_t, 276  
cmeif\_disable  
    vtss\_vstax\_conf\_t, 546  
comm\_opt  
    vtss\_phy\_ts\_ach\_conf\_t, 364  
    vtss\_phy\_ts\_eth\_conf\_t, 372  
    vtss\_phy\_ts\_gen\_conf\_t, 381  
    vtss\_phy\_ts\_ip\_conf\_t, 392  
    vtss\_phy\_ts\_mpls\_conf\_t, 395  
complete  
    vtss\_phy\_10g\_clause\_37\_status\_t, 254  
    vtss\_phy\_10g\_kr\_status\_aneg\_t, 272  
    vtss\_phy\_10g\_kr\_status\_train\_t, 276  
config\_bit\_mask  
    vtss\_phy\_10g\_ib\_conf\_t, 265  
connector\_enable  
    vtss\_phy\_loopback\_t, 343  
context  
    vtss\_ts\_timestamp\_alloc\_t, 509  
    vtss\_ts\_timestamp\_t, 510  
copper  
    vtss\_port\_status\_t, 460  
copy\_to\_cpu  
    vtss\_mac\_table\_entry\_t, 185  
corrected\_block\_cnt  
    vtss\_phy\_10g\_kr\_status\_fec\_t, 275  
cos  
    vtss\_packet\_rx\_info\_t, 206  
    vtss\_packet\_tx\_info\_t, 227  
cp\_ob\_tap\_result  
    vtss\_phy\_10g\_kr\_status\_train\_t, 276  
cpu

vtss\_acl\_action\_t, 60  
 vtss\_learn\_mode\_t, 183  
 cpu\_once  
     vtss\_acl\_action\_t, 60  
 cpu\_queue  
     vtss\_acl\_action\_t, 61  
     vtss\_mac\_table\_entry\_t, 185  
     vtss\_policer\_ext\_t, 424  
 cs\_wait\_ns  
     vtss\_pi\_conf\_t, 421  
 cu\_bad  
     vtss\_phy\_statistic\_t, 358  
 cu\_good  
     vtss\_phy\_statistic\_t, 358  
 cur\_version  
     vtss\_restart\_status\_t, 488  
 cw\_en  
     vtss\_phy\_ts\_mpls\_conf\_t, 395  
 d\_filter  
     vtss\_phy\_10g\_mode\_t, 286  
 d\_fltr  
     vtss\_phy\_10g\_ob\_status\_t, 294  
 dais\_l  
     vtss\_ewis\_defects\_s, 110  
 dais\_p  
     vtss\_ewis\_defects\_s, 110  
 data  
     vtss\_ace\_frame\_etype\_t, 43  
     vtss\_ace\_frame\_ipv4\_t, 46  
     vtss\_ace\_frame\_ipv6\_t, 50  
     vtss\_phy\_ts\_gen\_conf\_t, 382  
     vtss\_phy\_ts\_generic\_action\_t, 383  
     vtss\_qce\_frame\_etype\_t, 463  
     vtss\_qce\_frame\_llc\_t, 467  
     vtss\_qce\_frame\_snap\_t, 467  
     vtss\_vce\_frame\_etype\_t, 525  
     vtss\_vce\_frame\_llc\_t, 529  
     vtss\_vce\_frame\_snap\_t, 529  
 ddr\_mode  
     vtss\_phy\_10g\_mode\_t, 289  
 ddr\_mode\_t  
     vtss\_phy\_10g\_api.h, 865  
 default\_dei  
     vtss\_qos\_port\_conf\_t, 479  
 default\_dpl  
     vtss\_qos\_port\_conf\_t, 479  
 default\_prio  
     vtss\_qos\_port\_conf\_t, 479  
 dei  
     vtss\_ece\_inner\_tag\_t, 81  
     vtss\_ece\_outer\_tag\_t, 86  
     vtss\_ece\_tag\_t, 88  
     vtss\_mirror\_conf\_t, 190  
     vtss\_qce\_tag\_t, 474  
     vtss\_vce\_tag\_t, 535  
     vtss\_vlan\_tag\_t, 541  
 dei\_colouring  
     vtss\_evc\_port\_conf\_t, 100  
 delaym\_type  
     vtss\_phy\_ts\_ietf\_mpls\_ach\_oam\_conf\_t, 386  
     vtss\_phy\_ts\_ptp\_engine\_action\_t, 408  
     vtss\_phy\_ts\_y1731\_oam\_conf\_t, 414  
 des\_interface\_width  
     vtss\_phy\_10g\_prbs\_mon\_conf\_t, 307  
 dest\_ip  
     vtss\_phy\_ts\_fifo\_sig\_t, 380  
 dest\_mac  
     vtss\_phy\_ts\_fifo\_sig\_t, 380  
 destination  
     vtss\_ipv4\_uc\_t, 168  
     vtss\_ipv6\_uc\_t, 170  
     vtss\_irq\_conf\_t, 171  
     vtss\_mac\_table\_entry\_t, 185  
 detect\_only  
     vtss\_phy\_ts\_fifo\_conf\_t, 377  
 device\_feature\_status  
     vtss\_phy\_10g\_id\_t, 269  
 dfais\_p  
     vtss\_ewis\_defects\_s, 111  
 dfe1  
     vtss\_phy\_10g\_ib\_conf\_t, 263  
 dfe2  
     vtss\_phy\_10g\_ib\_conf\_t, 263  
 dfe3  
     vtss\_phy\_10g\_ib\_conf\_t, 264  
 dfe4  
     vtss\_phy\_10g\_ib\_conf\_t, 264  
 dfplm\_p  
     vtss\_ewis\_defects\_s, 111  
 dfuneq\_p  
     vtss\_ewis\_defects\_s, 112  
 dgroup\_no  
     vtss\_dgroup\_port\_conf\_t, 74  
 dig\_offset\_reg  
     vtss\_phy\_10g\_mode\_t, 285  
 dip  
     vtss\_ace\_frame\_arp\_t, 42  
     vtss\_ace\_frame\_ipv4\_t, 46  
     vtss\_l3\_neighbour\_t, 177  
 dir  
     vtss\_ece\_action\_t, 77  
 disable  
     vtss\_phy\_mac\_serdes\_pcs\_ctrl\_t, 346  
 discard  
     vtss\_learn\_mode\_t, 183  
 divider  
     vtss\_sync\_clock\_out\_t, 500  
 dlcd\_p  
     vtss\_ewis\_defects\_s, 111  
 dllof\_s  
     vtss\_ewis\_defects\_s, 109  
 dllop\_p  
     vtss\_ewis\_defects\_s, 110  
 dlos\_s  
     vtss\_ewis\_defects\_s, 109  
 dma\_enable

vtss\_packet\_dma\_conf\_t, 193  
dmac  
  vtss\_ace\_frame\_etype\_t, 43  
  vtss\_ace\_frame\_llc\_t, 53  
  vtss\_ace\_frame\_snap\_t, 54  
  vtss\_ece\_mac\_t, 84  
  vtss\_l3\_neighbour\_t, 177  
  vtss\_phy\_10g\_pkt\_gen\_conf\_t, 299  
dmac\_bc  
  vtss\_ace\_t, 57  
  vtss\_qce\_mac\_t, 471  
  vtss\_vce\_mac\_t, 533  
dmac\_dip  
  vtss\_vcl\_port\_conf\_t, 536  
dmac\_enable  
  vtss\_aggr\_mode\_t, 65  
dmac\_match  
  vtss\_ace\_frame\_arp\_t, 41  
dmac\_mc  
  vtss\_ace\_t, 56  
  vtss\_qce\_mac\_t, 471  
  vtss\_vce\_mac\_t, 533  
domain  
  vtss\_phy\_ts\_ptp\_conf\_t, 406  
domain\_num  
  vtss\_phy\_ts\_fifo\_sig\_t, 379  
doof\_s  
  vtss\_ewis\_defects\_s, 109  
dot1dTpPortInDiscards  
  vtss\_port\_bridge\_counters\_t, 426  
dot3ControlInUnknownOpcodes  
  vtss\_port\_ethernet\_like\_counters\_t, 436  
dot3InPauseFrames  
  vtss\_port\_ethernet\_like\_counters\_t, 436  
dot3OutPauseFrames  
  vtss\_port\_ethernet\_like\_counters\_t, 438  
dot3StatsAlignmentErrors  
  vtss\_port\_ethernet\_like\_counters\_t, 435  
dot3StatsCarrierSenseErrors  
  vtss\_port\_ethernet\_like\_counters\_t, 438  
dot3StatsDeferredTransmissions  
  vtss\_port\_ethernet\_like\_counters\_t, 437  
dot3StatsExcessiveCollisions  
  vtss\_port\_ethernet\_like\_counters\_t, 437  
dot3StatsFCSErrors  
  vtss\_port\_ethernet\_like\_counters\_t, 436  
dot3StatsFrameTooLongs  
  vtss\_port\_ethernet\_like\_counters\_t, 436  
dot3StatsLateCollisions  
  vtss\_port\_ethernet\_like\_counters\_t, 437  
dot3StatsMultipleCollisionFrames  
  vtss\_port\_ethernet\_like\_counters\_t, 437  
dot3StatsSingleCollisionFrames  
  vtss\_port\_ethernet\_like\_counters\_t, 437  
dot3StatsSymbolErrors  
  vtss\_port\_ethernet\_like\_counters\_t, 436  
dp  
  vtss\_packet\_tx\_info\_t, 232  
                vtss\_qce\_action\_t, 461  
                vtss\_vstax\_tx\_header\_t, 554  
dp\_bypass\_level  
  vtss\_policer\_ext\_t, 422  
dp\_enable  
  vtss\_qce\_action\_t, 461  
dp\_level\_map  
  vtss\_qos\_port\_conf\_t, 480  
dplm\_p  
  vtss\_ewis\_defects\_s, 111  
dport  
  vtss\_ace\_frame\_ipv4\_t, 46  
  vtss\_ace\_frame\_ipv6\_t, 50  
  vtss\_qce\_frame\_ipv4\_t, 464  
  vtss\_qce\_frame\_ipv6\_t, 466  
  vtss\_vce\_frame\_ipv4\_t, 526  
  vtss\_vce\_frame\_ipv6\_t, 528  
dport\_mask  
  vtss\_phy\_ts\_ip\_conf\_t, 392  
dport\_val  
  vtss\_phy\_ts\_ip\_conf\_t, 392  
drdi\_l  
  vtss\_ewis\_defects\_s, 110  
drdi\_p  
  vtss\_ewis\_defects\_s, 111  
ds  
  vtss\_ace\_frame\_ipv4\_t, 45  
  vtss\_ace\_frame\_ipv6\_t, 50  
  vtss\_phy\_ts\_ietf\_mpls\_ach\_oam\_conf\_t, 386  
dscp  
  vtss\_ece\_frame\_ipv4\_t, 79  
  vtss\_ece\_frame\_ipv6\_t, 80  
  vtss\_qce\_action\_t, 462  
  vtss\_qce\_frame\_ipv4\_t, 464  
  vtss\_qce\_frame\_ipv6\_t, 465  
  vtss\_vce\_frame\_ipv4\_t, 526  
  vtss\_vce\_frame\_ipv6\_t, 527  
dscp\_class\_enable  
  vtss\_qos\_port\_conf\_t, 480  
dscp\_dp\_level\_map  
  vtss\_qos\_conf\_t, 475  
dscp\_emode  
  vtss\_qos\_port\_conf\_t, 480  
dscp\_enable  
  vtss\_qce\_action\_t, 462  
dscp\_mode  
  vtss\_qos\_port\_conf\_t, 480  
dscp\_qos\_class\_map  
  vtss\_qos\_conf\_t, 475  
dscp\_qos\_map  
  vtss\_qos\_conf\_t, 475  
dscp\_remap  
  vtss\_qos\_conf\_t, 476  
dscp\_remark  
  vtss\_qos\_conf\_t, 476  
dscp\_translate  
  vtss\_qos\_port\_conf\_t, 481  
dscp\_translate\_map

vtss\_qos\_conf\_t, 476  
 dscp\_trust  
     vtss\_qos\_conf\_t, 475  
 dst\_port\_mask  
     vtss\_packet\_tx\_info\_t, 224  
 dual\_media\_fiber\_speed  
     port\_custom\_conf\_t, 34  
 duneq\_p  
     vtss\_ewis\_defects\_s, 110  
 dwrr\_enable  
     vtss\_qos\_port\_conf\_t, 482  
  
 ebs  
     vtss\_dlb\_policer\_conf\_t, 76  
 edc\_fw\_api\_load  
     vtss\_phy\_10g\_fw\_status\_t, 259  
 edc\_fw\_chksum  
     vtss\_phy\_10g\_fw\_status\_t, 258  
 edc\_fw\_load  
     vtss\_phy\_10g\_mode\_t, 284  
 edc\_fw\_rev  
     vtss\_phy\_10g\_fw\_status\_t, 258  
 eee\_ena  
     vtss\_eee\_port\_conf\_t, 89  
 eee\_ena\_phy  
     vtss\_phy\_eee\_conf\_t, 339  
 eee\_fast\_queues  
     vtss\_eee\_port\_conf\_t, 89  
 eee\_mode  
     vtss\_phy\_eee\_conf\_t, 338  
 egr\_fcs\_err  
     vtss\_phy\_ts\_stats\_t, 412  
 egr\_frm\_mod\_cnt  
     vtss\_phy\_ts\_stats\_t, 413  
 egr\_pream\_shrink\_err  
     vtss\_phy\_ts\_stats\_t, 412  
 eir  
     vtss\_dlb\_policer\_conf\_t, 76  
 en\_ob  
     vtss\_phy\_10g\_base\_kr\_conf\_t, 241  
 ena\_ifh\_header  
     vtss\_port\_ifh\_t, 445  
 enable  
     port\_custom\_conf\_t, 33  
     vtss\_dlb\_policer\_conf\_t, 75  
     vtss\_ece\_outer\_tag\_t, 85  
     vtss\_fdma\_cfg\_t, 142  
     vtss\_mac\_table\_entry\_t, 186  
     vtss\_npi\_conf\_t, 191  
     vtss\_packet\_rx\_queue\_npi\_conf\_t, 220  
     vtss\_phy\_10g\_auto\_failover\_conf\_t, 238  
     vtss\_phy\_10g\_base\_kr\_training\_t, 246  
     vtss\_phy\_10g\_ckout\_conf\_t, 249  
     vtss\_phy\_10g\_clause\_37\_control\_t, 253  
     vtss\_phy\_10g\_kr\_status\_fec\_t, 275  
     vtss\_phy\_10g\_lane\_sync\_conf\_t, 277  
     vtss\_phy\_10g\_loopback\_t, 280  
     vtss\_phy\_10g\_pkt\_gen\_conf\_t, 297  
     vtss\_phy\_10g\_pkt\_mon\_conf\_t, 301  
  
     vtss\_phy\_10g\_prbs\_gen\_conf\_t, 304  
     vtss\_phy\_10g\_prbs\_mon\_conf\_t, 306  
     vtss\_phy\_10g\_sckout\_conf\_t, 312  
     vtss\_phy\_10g\_srefclk\_mode\_t, 320  
     vtss\_phy\_10g\_vscope\_conf\_t, 325  
     vtss\_phy\_ltc\_freq\_synth\_s, 345  
     vtss\_phy\_ts\_generic\_action\_t, 383  
     vtss\_phy\_ts\_nphase\_status\_t, 399  
     vtss\_phy\_ts\_oam\_engine\_action\_t, 401  
     vtss\_phy\_ts\_ptp\_engine\_action\_t, 407  
     vtss\_port\_clause\_37\_control\_t, 428  
     vtss\_red\_t, 486  
     vtss\_sync\_clock\_in\_t, 499  
     vtss\_sync\_clock\_out\_t, 500  
     vtss\_ts\_ext\_clock\_mode\_t, 505  
 enable\_pass\_thru  
     vtss\_phy\_10g\_clause\_37\_control\_t, 253  
     vtss\_phy\_10g\_mode\_t, 292  
 enabled  
     vtss\_sgpi\_port\_conf\_t, 495  
 encaps\_type  
     vtss\_phy\_ts\_eng\_init\_conf\_t, 366  
 end  
     vtss\_phy\_ts\_mpls\_conf\_t, 397  
 eng\_minE  
     vtss\_phy\_ts\_fifo\_conf\_t, 378  
 eng\_mode  
     vtss\_phy\_ts\_engine\_flow\_conf\_t, 369  
 eng\_recov  
     vtss\_phy\_ts\_fifo\_conf\_t, 377  
 eng\_used  
     vtss\_phy\_ts\_eng\_init\_conf\_t, 366  
 err\_blk\_cnt  
     vtss\_phy\_pcs\_cnt\_t, 352  
 error\_counter  
     vtss\_phy\_10g\_pcs\_prbs\_mon\_conf\_t, 296  
 error\_free\_x  
     vtss\_phy\_10g\_vscope\_scan\_status\_t, 329  
 error\_free\_y  
     vtss\_phy\_10g\_vscope\_scan\_status\_t, 329  
 error\_states  
     vtss\_phy\_10g\_prbs\_mon\_conf\_t, 307  
 error\_status  
     vtss\_phy\_10g\_prbs\_mon\_conf\_t, 308  
 error\_thres  
     vtss\_phy\_10g\_vscope\_conf\_t, 325  
 errors  
     vtss\_phy\_10g\_vscope\_scan\_status\_t, 329  
 eth1\_opt  
     vtss\_phy\_ts\_generic\_flow\_conf\_t, 385  
     vtss\_phy\_ts\_oam\_engine\_flow\_conf\_t, 402  
     vtss\_phy\_ts\_ptp\_engine\_flow\_conf\_t, 409  
 eth2\_opt  
     vtss\_phy\_ts\_oam\_engine\_flow\_conf\_t, 403  
     vtss\_phy\_ts\_ptp\_engine\_flow\_conf\_t, 409  
 ethernet  
     vtss\_ace\_frame\_arp\_t, 42  
 ethernet\_like

vtss\_port\_counters\_t, 434  
etype  
  vtss\_ace\_frame\_etype\_t, 43  
  vtss\_ace\_t, 57  
  vtss\_packet\_rx\_meta\_t, 213  
  vtss\_phy\_10g\_pkt\_gen\_conf\_t, 298  
  vtss\_phy\_ts\_eth\_conf\_t, 372  
  vtss\_qce\_frame\_etype\_t, 463  
  vtss\_qce\_key\_t, 469  
  vtss\_vce\_frame\_etype\_t, 524  
  vtss\_vce\_key\_t, 531  
evc\_id  
  vtss\_ece\_action\_t, 78  
evnt  
  vtss\_phy\_10g\_auto\_failover\_conf\_t, 237  
ewis\_cnt\_cfg  
  vtss\_ewis\_static\_conf\_s, 128  
ewis\_cntr\_thresh\_conf  
  vtss\_ewis\_conf\_s, 104  
ewis\_init\_done  
  vtss\_ewis\_conf\_s, 101  
ewis\_lof\_ctrl1  
  vtss\_ewis\_static\_conf\_s, 127  
ewis\_lof\_ctrl2  
  vtss\_ewis\_static\_conf\_s, 127  
ewis\_mode  
  vtss\_ewis\_conf\_s, 102  
ewis\_mode\_ctrl  
  vtss\_ewis\_static\_conf\_s, 126  
ewis\_pmtick\_ctrl  
  vtss\_ewis\_static\_conf\_s, 128  
ewis\_rx\_ctrl1  
  vtss\_ewis\_static\_conf\_s, 125  
ewis\_rx\_err\_frc1  
  vtss\_ewis\_static\_conf\_s, 127  
ewis\_rx\_frm\_ctrl1  
  vtss\_ewis\_static\_conf\_s, 127  
ewis\_tx\_a1\_a2  
  vtss\_ewis\_static\_conf\_s, 126  
ewis\_tx\_c2\_h1  
  vtss\_ewis\_static\_conf\_s, 126  
ewis\_tx\_h2\_h3  
  vtss\_ewis\_static\_conf\_s, 126  
ewis\_tx\_z0\_e1  
  vtss\_ewis\_static\_conf\_s, 126  
ewis\_txctrl1  
  vtss\_ewis\_static\_conf\_s, 125  
ewis\_txctrl2  
  vtss\_ewis\_static\_conf\_s, 125  
exc\_col\_cont  
  port\_custom\_conf\_t, 34  
  vtss\_port\_conf\_t, 432  
excess\_enable  
  vtss\_qos\_port\_conf\_t, 478  
exp\_sl  
  vtss\_ewis\_conf\_s, 103  
exsl  
  vtss\_ewis\_sl\_conf\_s, 124  
external  
  vtss\_irq\_conf\_t, 171  
f\_ebc\_thr\_l  
  vtss\_ewis\_counter\_threshold\_s, 108  
f\_ebc\_thr\_p  
  vtss\_ewis\_counter\_threshold\_s, 108  
FALSE  
  types.h, 602  
family  
  vtss\_phy\_10g\_id\_t, 269  
fan\_low\_pol  
  vtss\_fan\_conf\_t, 140  
fan\_open\_col  
  vtss\_fan\_conf\_t, 141  
fan\_pwm\_freq  
  vtss\_fan\_conf\_t, 140  
far\_end\_enable  
  vtss\_phy\_loopback\_t, 343  
fast\_link\_stat\_ena  
  vtss\_phy\_mac\_serdes\_pcs\_ctrl\_t, 348  
fault  
  vtss\_ewis\_cons\_act\_s, 105  
  vtss\_ewis\_status\_s, 129  
fault\_on\_aisl  
  vtss\_ewis\_fault\_cons\_act\_s, 114  
fault\_on\_aisp  
  vtss\_ewis\_fault\_cons\_act\_s, 114  
fault\_on\_feaisp  
  vtss\_ewis\_fault\_cons\_act\_s, 113  
fault\_on\_fepImp  
  vtss\_ewis\_fault\_cons\_act\_s, 113  
fault\_on\_lcdp  
  vtss\_ewis\_fault\_cons\_act\_s, 114  
fault\_on\_lof  
  vtss\_ewis\_fault\_cons\_act\_s, 113  
fault\_on\_lopp  
  vtss\_ewis\_fault\_cons\_act\_s, 114  
fault\_on\_los  
  vtss\_ewis\_fault\_cons\_act\_s, 113  
fault\_on\_plmp  
  vtss\_ewis\_fault\_cons\_act\_s, 114  
fault\_on\_rdil  
  vtss\_ewis\_fault\_cons\_act\_s, 113  
fault\_on\_sef  
  vtss\_ewis\_fault\_cons\_act\_s, 113  
fcs  
  vtss\_packet\_rx\_meta\_t, 213  
fdx  
  port\_custom\_conf\_t, 33  
  vtss\_phy\_10g\_clause\_37\_adv\_t, 250  
  vtss\_phy\_forced\_t, 341  
  vtss\_port\_clause\_37\_adv\_t, 427  
  vtss\_port\_conf\_t, 431  
  vtss\_port\_sgmii\_aneg\_t, 456  
  vtss\_port\_status\_t, 459  
fdx\_gap

vtss\_port\_frame\_gaps\_t, 440  
 fec  
     vtss\_phy\_10g\_base\_kr\_status\_t, 244  
 fec\_abil  
     vtss\_phy\_10g\_base\_kr\_id\_adv\_abil\_t, 243  
 fec\_enable  
     vtss\_phy\_10g\_kr\_status\_aneg\_t, 273  
 fec\_req  
     vtss\_phy\_10g\_base\_kr\_id\_adv\_abil\_t, 243  
 fiber  
     vtss\_port\_status\_t, 460  
 fid  
     vtss\_vlan\_vid\_conf\_t, 545  
 file  
     vtss\_api\_lock\_t, 67  
 fill\_level  
     vtss\_eee\_port\_counter\_t, 91  
 fill\_level\_get  
     vtss\_eee\_port\_counter\_t, 91  
 fill\_level\_thres  
     vtss\_eee\_port\_counter\_t, 91  
 filter  
     vtss\_packet\_port\_filter\_t, 196  
     vtss\_phy\_10g\_auto\_failover\_conf\_t, 238  
 flf  
     vtss\_phy\_conf\_t, 336  
 flooded  
     vtss\_policer\_ext\_t, 423  
 flow\_conf  
     vtss\_phy\_ts\_engine\_flow\_conf\_t, 370  
 flow\_control  
     port\_custom\_conf\_t, 33  
     vtss\_policer\_ext\_t, 424  
     vtss\_port\_conf\_t, 431  
 flow\_en  
     vtss\_phy\_ts\_eth\_conf\_t, 372  
     vtss\_phy\_ts\_gen\_conf\_t, 381  
     vtss\_phy\_ts\_ip\_conf\_t, 392  
     vtss\_phy\_ts\_mpls\_conf\_t, 395  
 flow\_end\_index  
     vtss\_phy\_ts\_eng\_init\_conf\_t, 367  
 flow\_id  
     vtss\_phy\_ts\_generic\_action\_t, 383  
 flow\_match\_mode  
     vtss\_phy\_ts\_eng\_init\_conf\_t, 366  
 flow\_offset  
     vtss\_phy\_ts\_gen\_conf\_t, 381  
 flow\_opt  
     vtss\_phy\_ts\_eth\_conf\_t, 376  
     vtss\_phy\_ts\_gen\_conf\_t, 382  
     vtss\_phy\_ts\_ip\_conf\_t, 394  
     vtss\_phy\_ts\_mpls\_conf\_t, 398  
 flow\_st\_index  
     vtss\_phy\_ts\_eng\_init\_conf\_t, 366  
 fltr\_val  
     vtss\_phy\_10g\_auto\_failover\_conf\_t, 238  
 force  
     vtss\_phy\_reset\_conf\_t, 355  
 force\_adv\_ability  
     vtss\_phy\_mac\_serdes\_pcs\_ctrl\_t, 347  
     vtss\_phy\_media\_serdes\_pcs\_ctrl\_t, 350  
 force\_ais  
     vtss\_ewis\_line\_force\_mode\_s, 116  
     vtss\_ewis\_line\_tx\_force\_mode\_s, 117  
 force\_ams\_sel  
     vtss\_phy\_conf\_t, 337  
 force\_fefi  
     vtss\_phy\_media\_serdes\_pcs\_ctrl\_t, 351  
 force\_fefi\_value  
     vtss\_phy\_media\_serdes\_pcs\_ctrl\_t, 351  
 force\_hls  
     vtss\_phy\_media\_serdes\_pcs\_ctrl\_t, 351  
 force\_mode  
     vtss\_ewis\_conf\_s, 102  
 force\_rdi  
     vtss\_ewis\_line\_force\_mode\_s, 117  
     vtss\_ewis\_line\_tx\_force\_mode\_s, 118  
     vtss\_ewis\_path\_force\_mode\_s, 119  
 force\_uneq  
     vtss\_ewis\_path\_force\_mode\_s, 118  
 forced  
     vtss\_phy\_conf\_t, 335  
 forward  
     vtss\_acl\_action\_t, 61  
 frag  
     vtss\_phy\_10g\_pkt\_mon\_conf\_t, 302  
 fragment  
     vtss\_ace\_frame\_ipv4\_t, 45  
     vtss\_qce\_frame\_ipv4\_t, 464  
     vtss\_vce\_frame\_ipv4\_t, 525  
 frame  
     vtss\_ace\_t, 58  
     vtss\_ece\_key\_t, 83  
     vtss\_qce\_key\_t, 470  
     vtss\_vce\_key\_t, 532  
 frame\_gaps  
     vtss\_port\_conf\_t, 430  
 frame\_length\_chk  
     port\_custom\_conf\_t, 35  
     vtss\_port\_conf\_t, 432  
 frame\_rate  
     vtss\_policer\_ext\_t, 422  
 frame\_single  
     vtss\_phy\_10g\_pkt\_gen\_conf\_t, 298  
 frame\_type  
     vtss\_vlan\_port\_conf\_t, 540  
 frames  
     vtss\_counter\_pair\_t, 70  
     vtss\_phy\_10g\_pkt\_gen\_conf\_t, 298  
 freeze  
     vtss\_phy\_10g\_apc\_status\_t, 236  
 freeze\_bit\_mask  
     vtss\_phy\_10g\_ib\_conf\_t, 265  
 freq  
     vtss\_phy\_10g\_ckout\_conf\_t, 248  
     vtss\_phy\_10g\_sckout\_conf\_t, 312

vtss\_phy\_10g\_srefclk\_mode\_t, 320  
vtss\_phy\_clock\_conf\_t, 333  
vtss\_ts\_ext\_clock\_mode\_t, 506  
frm\_len  
    vtss\_packet\_tx\_info\_t, 225  
frm\_limit\_per\_tick  
    vtss\_fdma\_throttle\_cfg\_t, 151  
frm\_ptr  
    tag\_vtss\_fdma\_list, 37  
frst\_lvl\_after\_top  
    vtss\_phy\_ts\_mpls\_conf\_t, 396  
frst\_lvl\_before\_end  
    vtss\_phy\_ts\_mpls\_conf\_t, 397  
fsm\_lock  
    vtss\_lcpoll\_status\_t, 181  
fsm\_stat  
    vtss\_lcpoll\_status\_t, 181  
full  
    vtss\_debug\_info\_t, 72  
function  
    vtss\_api\_lock\_t, 67  
fwd\_enable  
    vtss\_mirror\_conf\_t, 189  
fwd\_mode  
    vtss\_vstax\_tx\_header\_t, 553  
gain  
    vtss\_phy\_10g\_ib\_conf\_t, 262  
gain\_stat  
    vtss\_lcpoll\_status\_t, 182  
gainadj  
    vtss\_phy\_10g\_ib\_conf\_t, 262  
garp\_cpu\_only  
    vtss\_packet\_rx\_reg\_t, 221  
garp\_queue  
    vtss\_packet\_rx\_queue\_map\_t, 218  
garp\_reg  
    vtss\_packet\_rx\_port\_conf\_t, 216  
gen  
    vtss\_phy\_ts\_engine\_flow\_conf\_t, 370  
gen\_conf  
    vtss\_phy\_ts\_engine\_action\_t, 368  
gen\_opt  
    vtss\_phy\_ts\_generic\_flow\_conf\_t, 385  
generate  
    vtss\_port\_flow\_control\_conf\_t, 439  
generate\_pause  
    vtss\_aneg\_t, 66  
glag\_no  
    vtss\_packet\_frame\_info\_t, 194  
    vtss\_packet\_port\_info\_t, 197  
    vtss\_packet\_rx\_info\_t, 203  
    vtss\_vstax\_rx\_header\_t, 551  
    vtss\_vstax\_tx\_header\_t, 554  
good\_crc  
    vtss\_phy\_10g\_pkt\_mon\_conf\_t, 301  
group  
    vtss\_debug\_info\_t, 71  
group\_id  
    vtss\_vlan\_trans\_grp2vlan\_conf\_t, 542  
    vtss\_vlan\_trans\_port2grp\_conf\_t, 543  
grp\_map  
    vtss\_packet\_rx\_conf\_t, 199  
h\_apc\_conf  
    vtss\_phy\_10g\_mode\_t, 291  
h\_clk\_src  
    vtss\_phy\_10g\_mode\_t, 290  
h\_ib\_conf  
    vtss\_phy\_10g\_mode\_t, 291  
h\_media  
    vtss\_phy\_10g\_mode\_t, 291  
h\_offset\_guard  
    vtss\_phy\_10g\_mode\_t, 288  
h\_pcs  
    vtss\_phy\_10g\_serdes\_status\_t, 319  
h\_pll5g\_fsm\_lock  
    vtss\_phy\_10g\_serdes\_status\_t, 314  
h\_pll5g\_fsm\_stat  
    vtss\_phy\_10g\_serdes\_status\_t, 314  
h\_pll5g\_gain  
    vtss\_phy\_10g\_serdes\_status\_t, 314  
h\_pll5g\_lock\_status  
    vtss\_phy\_10g\_serdes\_status\_t, 314  
h\_pma  
    vtss\_phy\_10g\_serdes\_status\_t, 318  
h\_rx\_rcpll\_fsm\_status  
    vtss\_phy\_10g\_serdes\_status\_t, 316  
h\_rx\_rcpll\_lock\_status  
    vtss\_phy\_10g\_serdes\_status\_t, 315  
h\_rx\_rcpll\_range  
    vtss\_phy\_10g\_serdes\_status\_t, 315  
h\_rx\_rcpll\_vco\_load  
    vtss\_phy\_10g\_serdes\_status\_t, 316  
h\_tx\_rcpll\_fsm\_status  
    vtss\_phy\_10g\_serdes\_status\_t, 317  
h\_tx\_rcpll\_lock\_status  
    vtss\_phy\_10g\_serdes\_status\_t, 317  
h\_tx\_rcpll\_range  
    vtss\_phy\_10g\_serdes\_status\_t, 317  
h\_tx\_rcpll\_vco\_load  
    vtss\_phy\_10g\_serdes\_status\_t, 317  
hdx  
    vtss\_phy\_10g\_clause\_37\_adv\_t, 250  
    vtss\_port\_clause\_37\_adv\_t, 427  
    vtss\_port\_sgmii\_aneg\_t, 456  
hdx\_gap\_1  
    vtss\_port\_frame\_gaps\_t, 440  
hdx\_gap\_2  
    vtss\_port\_frame\_gaps\_t, 440  
high  
    vtss\_phy\_timestamp\_t, 361  
    vtss\_vcap\_udp\_tcp\_t, 520  
    vtss\_vcap\_vr\_t, 522  
high\_ber\_latched  
    vtss\_phy\_pcs\_cnt\_t, 352  
high\_duty\_cycle  
    vtss\_phy\_ltc\_freq\_synth\_s, 345

high\_input\_gain  
     vtss\_phy\_10g\_mode\_t, 283

hints  
     vtss\_packet\_rx\_info\_t, 202

hl\_clk\_synth  
     vtss\_phy\_10g\_mode\_t, 283

host  
     vtss\_phy\_10g\_clause\_37\_cmn\_status\_t, 252  
     vtss\_phy\_10g\_clause\_37\_control\_t, 253

host\_kr  
     vtss\_phy\_10g\_base\_kr\_train\_aneg\_t, 245

host\_rx  
     vtss\_phy\_10g\_polarity\_inv\_t, 303

host\_tx  
     vtss\_phy\_10g\_polarity\_inv\_t, 303

hpccs  
     vtss\_phy\_10g\_status\_t, 322

hpccs\_1g  
     vtss\_phy\_10g\_status\_t, 322

hpma  
     vtss\_phy\_10g\_status\_t, 321

hw\_cnt  
     vtss\_os\_timestamp\_t, 192

hw\_tstamp  
     vtss\_packet\_rx\_info\_t, 208

hw\_tstamp\_decoded  
     vtss\_packet\_rx\_info\_t, 208

i16  
     types.h, 619

i32  
     types.h, 619

i64  
     types.h, 619

i8  
     types.h, 619

i\_cpu\_en  
     vtss\_phy\_reset\_conf\_t, 356

i\_tag  
     vtss\_phy\_ts\_eth\_conf\_t, 376

ib\_conf  
     vtss\_phy\_10g\_ib\_status\_t, 266  
     vtss\_phy\_10g\_mode\_t, 285

ib\_cterm\_ena  
     vtss\_serdess\_macro\_conf\_t, 492

ib\_ini\_lp  
     vtss\_phy\_10g\_mode\_t, 286

ib\_max\_lp  
     vtss\_phy\_10g\_mode\_t, 287

ib\_min\_lp  
     vtss\_phy\_10g\_mode\_t, 287

ib\_par\_cfg, 31  
     max, 32  
     min, 31  
     value, 31

ib\_storage  
     vtss\_phy\_10g\_ib\_storage\_t, 267

ib\_storage\_bool  
     vtss\_phy\_10g\_ib\_storage\_t, 267

icpu\_activity  
     vtss\_phy\_10g\_fw\_status\_t, 259

id  
     vtss\_ace\_t, 55  
     vtss\_ece\_t, 87  
     vtss\_qce\_t, 472  
     vtss\_ts\_timestamp\_t, 510  
     vtss\_vce\_t, 534

ietf\_oam\_conf  
     vtss\_phy\_ts\_oam\_engine\_action\_t, 401

if\_group  
     vtss\_port\_counters\_t, 434

if\_type  
     vtss\_port\_conf\_t, 430

ifInBroadcastPkts  
     vtss\_port\_if\_group\_counters\_t, 442

ifInDiscards  
     vtss\_port\_if\_group\_counters\_t, 442

ifInErrors  
     vtss\_port\_if\_group\_counters\_t, 443

ifInMulticastPkts  
     vtss\_port\_if\_group\_counters\_t, 442

ifInNUcastPkts  
     vtss\_port\_if\_group\_counters\_t, 442

ifInOctets  
     vtss\_port\_if\_group\_counters\_t, 441

ifInUcastPkts  
     vtss\_port\_if\_group\_counters\_t, 442

ifOutBroadcastPkts  
     vtss\_port\_if\_group\_counters\_t, 443

ifOutDiscards  
     vtss\_port\_if\_group\_counters\_t, 444

ifOutErrors  
     vtss\_port\_if\_group\_counters\_t, 444

ifOutMulticastPkts  
     vtss\_port\_if\_group\_counters\_t, 443

ifOutNUcastPkts  
     vtss\_port\_if\_group\_counters\_t, 444

ifOutOctets  
     vtss\_port\_if\_group\_counters\_t, 443

ifOutUcastPkts  
     vtss\_port\_if\_group\_counters\_t, 443

ifh  
     vtss\_packet\_tx\_ifh\_t, 223

igmp\_cpu\_only  
     vtss\_packet\_rx\_reg\_t, 222

igmp\_queue  
     vtss\_packet\_rx\_queue\_map\_t, 218

in\_range  
     vtss\_vcap\_udp\_tcp\_t, 519

in\_sig  
     vtss\_gpio\_10g\_gpio\_mode\_t, 157

incomplete  
     vtss\_phy\_10g\_prbs\_mon\_conf\_t, 309

incr\_levn  
     vtss\_phy\_10g\_jitter\_conf\_t, 271

ingr\_fcs\_err  
     vtss\_phy\_ts\_stats\_t, 412

ingr\_frm\_mod\_cnt  
    vtss\_phy\_ts\_stats\_t, 413

ingr\_pream\_shrink\_err  
    vtss\_phy\_ts\_stats\_t, 412

ingress  
    vtss\_phy\_10g\_pkt\_gen\_conf\_t, 298

ingress\_filter  
    vtss\_vlan\_port\_conf\_t, 540

inhibit\_odd\_start  
    vtss\_phy\_mac\_serdes\_pcs\_ctrl\_t, 349  
    vtss\_phy\_media\_serdes\_pcs\_ctrl\_t, 350

inj\_grp\_mask  
    vtss\_fdma\_ch\_cfg\_t, 148

inner\_tag  
    vtss\_ece\_action\_t, 78  
    vtss\_ece\_key\_t, 83  
    vtss\_phy\_ts\_eth\_conf\_t, 376

inner\_tag\_type  
    vtss\_phy\_ts\_eth\_conf\_t, 374

input  
    vtss\_gpio\_10g\_gpio\_mode\_t, 156

inst  
    vtss\_api\_lock\_t, 67

instable  
    vtss\_phy\_10g\_prbs\_mon\_conf\_t, 309

int\_fmt  
    vtss\_ts\_internal\_mode\_t, 507

int\_pol\_high  
    vtss\_sgpio\_port\_conf\_t, 495

interface  
    vtss\_phy\_10g\_mode\_t, 282

invert\_output  
    vtss\_gpio\_10g\_gpio\_mode\_t, 158

ip  
    vtss\_ace\_frame\_arp\_t, 41

ip1\_opt  
    vtss\_phy\_ts\_ptp\_engine\_flow\_conf\_t, 409

ip2\_opt  
    vtss\_phy\_ts\_ptp\_engine\_flow\_conf\_t, 409

ip\_addr  
    vtss\_phy\_ts\_ip\_conf\_t, 394

ip\_enable  
    vtss\_phy\_ts\_sertod\_conf\_t, 411

ip\_mode  
    vtss\_phy\_ts\_ip\_conf\_t, 391

ipg\_len  
    vtss\_phy\_10g\_pkt\_gen\_conf\_t, 299

ipmc\_ctrl\_cpu\_copy  
    vtss\_packet\_rx\_reg\_t, 221

ipmc\_ctrl\_queue  
    vtss\_packet\_rx\_queue\_map\_t, 218

ipv4  
    vtss\_ace\_t, 58  
    vtss\_ece\_key\_t, 83  
    vtss\_ip\_addr\_t, 165  
    vtss\_phy\_ts\_ip\_conf\_t, 393  
    vtss\_qce\_key\_t, 470  
    vtss\_vce\_key\_t, 531

    ipv4\_icmp\_redirect\_enable  
        vtss\_l3\_rleg\_conf\_t, 178

    ipv4\_uc  
        vtss\_routing\_entry\_t, 489

    ipv4\_unicast\_enable  
        vtss\_l3\_rleg\_conf\_t, 178

    ipv4uc\_received\_frames  
        vtss\_l3\_counters\_t, 175

    ipv4uc\_received\_octets  
        vtss\_l3\_counters\_t, 175

    ipv4uc\_transmitted\_frames  
        vtss\_l3\_counters\_t, 176

    ipv4uc\_transmitted\_octets  
        vtss\_l3\_counters\_t, 175

    ipv6

    ipv6\_icmp\_redirect\_enable  
        vtss\_l3\_rleg\_conf\_t, 179

    ipv6\_uc  
        vtss\_routing\_entry\_t, 489

    ipv6\_unicast\_enable  
        vtss\_l3\_rleg\_conf\_t, 178

    ipv6uc\_received\_frames  
        vtss\_l3\_counters\_t, 175

    ipv6uc\_received\_octets  
        vtss\_l3\_counters\_t, 175

    ipv6uc\_transmitted\_frames  
        vtss\_l3\_counters\_t, 176

    ipv6uc\_transmitted\_octets  
        vtss\_l3\_counters\_t, 176

    irq\_trigger  
        vtss\_acl\_action\_t, 62

    is\_high\_amp  
        vtss\_phy\_10g\_clk\_src\_t, 256

    is\_host  
        vtss\_phy\_10g\_ib\_conf\_t, 265  
        vtss\_phy\_10g\_ob\_status\_t, 295

    is\_host\_side  
        vtss\_phy\_10g\_auto\_failover\_conf\_t, 237

    is\_host\_wan  
        vtss\_phy\_10g\_mode\_t, 290

    is\_init  
        vtss\_phy\_10g\_mode\_t, 292

    isdx  
        vtss\_packet\_rx\_info\_t, 210  
        vtss\_packet\_tx\_info\_t, 231  
        vtss\_vstax\_rx\_header\_t, 552

    isdx\_dont\_use  
        vtss\_packet\_tx\_info\_t, 232

    ivid  
        vtss\_evc\_pb\_conf\_t, 98

    keep\_ttl  
        vtss\_vstax\_tx\_header\_t, 554

key  
 vtss\_ece\_t, 87  
 vtss\_qce\_t, 472  
 vtss\_vce\_t, 534

I  
 vtss\_phy\_10g\_ib\_conf\_t, 263

I2\_types.h  
 vtss\_sfloop\_type\_t, 557

I3\_other\_queue  
 vtss\_packet\_rx\_queue\_map\_t, 219

I3\_uc\_queue  
 vtss\_packet\_rx\_queue\_map\_t, 219

I\_apc\_conf  
 vtss\_phy\_10g\_mode\_t, 292

I\_clk\_src  
 vtss\_phy\_10g\_mode\_t, 290

I\_h  
 vtss\_phy\_10g\_clause\_37\_control\_t, 253

I\_ib\_conf  
 vtss\_phy\_10g\_mode\_t, 291

I\_media  
 vtss\_phy\_10g\_mode\_t, 291

I\_offset\_guard  
 vtss\_phy\_10g\_mode\_t, 288

I\_pcs  
 vtss\_phy\_10g\_serdes\_status\_t, 319

I\_pll5g\_fsm\_lock  
 vtss\_phy\_10g\_serdes\_status\_t, 315

I\_pll5g\_fsm\_stat  
 vtss\_phy\_10g\_serdes\_status\_t, 315

I\_pll5g\_gain  
 vtss\_phy\_10g\_serdes\_status\_t, 315

I\_pll5g\_lock\_status  
 vtss\_phy\_10g\_serdes\_status\_t, 314

I\_pma  
 vtss\_phy\_10g\_serdes\_status\_t, 319

I\_rx\_rcpll\_fsm\_status  
 vtss\_phy\_10g\_serdes\_status\_t, 317

I\_rx\_rcpll\_lock\_status  
 vtss\_phy\_10g\_serdes\_status\_t, 316

I\_rx\_rcpll\_range  
 vtss\_phy\_10g\_serdes\_status\_t, 316

I\_rx\_rcpll\_vco\_load  
 vtss\_phy\_10g\_serdes\_status\_t, 316

I\_tx\_rcpll\_fsm\_status  
 vtss\_phy\_10g\_serdes\_status\_t, 318

I\_tx\_rcpll\_lock\_status  
 vtss\_phy\_10g\_serdes\_status\_t, 318

I\_tx\_rcpll\_range  
 vtss\_phy\_10g\_serdes\_status\_t, 318

I\_tx\_rcpll\_vco\_load  
 vtss\_phy\_10g\_serdes\_status\_t, 318

latch\_timestamp  
 vtss\_packet\_tx\_info\_t, 230

layer  
 vtss\_debug\_info\_t, 71

lb\_type  
 vtss\_phy\_10g\_loopback\_t, 280

vtss\_phy\_api.h, 956

Id  
 vtss\_phy\_10g\_ib\_conf\_t, 264

Id\_abil  
 vtss\_phy\_10g\_base\_kr\_train\_aneg\_t, 245

Id\_pre\_init  
 vtss\_phy\_10g\_base\_kr\_training\_t, 247

leaf  
 vtss\_evc\_pb\_conf\_t, 99

leaf\_ivid  
 vtss\_evc\_pb\_conf\_t, 98

leaf\_vid  
 vtss\_evc\_pb\_conf\_t, 99

learn  
 vtss\_acl\_action\_t, 61  
 vtss\_packet\_rx\_header\_t, 200

learn\_queue  
 vtss\_packet\_rx\_queue\_map\_t, 218

learned  
 vtss\_mac\_table\_status\_t, 187

learning  
 vtss\_evc\_conf\_t, 95  
 vtss\_policer\_ext\_t, 423  
 vtss\_vlan\_vid\_conf\_t, 544

led\_conf  
 vtss\_gpio\_10g\_gpio\_mode\_t, 158

length  
 vtss\_ace\_frame\_arp\_t, 41  
 vtss\_packet\_rx\_header\_t, 200  
 vtss\_packet\_rx\_info\_t, 203  
 vtss\_packet\_rx\_meta\_t, 214  
 vtss\_packet\_tx\_ifh\_t, 223  
 vtss\_phy\_veriphy\_result\_t, 418

level  
 vtss\_phy\_power\_status\_t, 354  
 vtss\_policer\_t, 425  
 vtss\_shaper\_t, 497  
 vtss\_trace\_conf\_t, 505

levn  
 vtss\_phy\_10g\_jitter\_conf\_t, 271  
 vtss\_phy\_10g\_ob\_status\_t, 294

Ifault  
 vtss\_phy\_10g\_pkt\_mon\_conf\_t, 302

limit\_cpu\_traffic  
 vtss\_policer\_ext\_t, 424

limit\_noncpu\_traffic  
 vtss\_policer\_ext\_t, 424

line  
 vtss\_api\_lock\_t, 67  
 vtss\_phy\_10g\_clause\_37\_cmn\_status\_t, 252  
 vtss\_phy\_10g\_clause\_37\_control\_t, 253  
 vtss\_phy\_10g\_prbs\_gen\_conf\_t, 305  
 vtss\_phy\_10g\_prbs\_mon\_conf\_t, 306  
 vtss\_phy\_10g\_vsphere\_conf\_t, 325  
 vtss\_phy\_10g\_vsphere\_scan\_conf\_t, 326

line\_kr  
 vtss\_phy\_10g\_base\_kr\_train\_aneg\_t, 245

line\_rate

vtss\_dlb\_policer\_conf\_t, 75  
line\_rx  
    vtss\_phy\_10g\_polarity\_inv\_t, 303  
line\_rx\_force  
    vtss\_ewis\_force\_mode\_s, 115  
line\_tx  
    vtss\_phy\_10g\_polarity\_inv\_t, 303  
line\_tx\_force  
    vtss\_ewis\_force\_mode\_s, 115  
link  
    vtss\_phy\_10g\_clause\_37\_status\_t, 254  
    vtss\_phy\_veriphy\_result\_t, 418  
    vtss\_port\_sgmii\_aneg\_t, 456  
    vtss\_port\_status\_t, 458  
link\_6g\_distance  
    vtss\_phy\_10g\_mode\_t, 290  
link\_down  
    vtss\_port\_status\_t, 458  
    vtss\_sublayer\_status\_t, 498  
link\_stat  
    vtss\_ewis\_status\_s, 129  
list  
    vtss\_fdma\_ch\_cfg\_t, 148  
llabs  
    vtss\_os\_ecos.h, 800  
llc  
    vtss\_ace\_frame\_llc\_t, 53  
    vtss\_ace\_t, 57  
    vtss\_qce\_key\_t, 469  
    vtss\_vce\_key\_t, 531  
lock\_status  
    vtss\_lcpll\_status\_t, 181  
locked  
    vtss\_mac\_table\_entry\_t, 185  
loop  
    vtss\_port\_conf\_t, 433  
loopback  
    vtss\_ewis\_test\_conf\_s, 130  
lopc\_stat  
    vtss\_phy\_10g\_status\_t, 323  
low  
    vtss\_phy\_timestamp\_t, 361  
    vtss\_vcap\_udp\_tcp\_t, 519  
    vtss\_vcap\_vr\_t, 522  
low\_duty\_cycle  
    vtss\_phy\_ltc\_freq\_synth\_s, 345  
lower  
    vtss\_phy\_ts\_eth\_conf\_t, 374  
    vtss\_phy\_ts\_mpls\_lvl\_rng\_t, 398  
    vtss\_phy\_ts\_ptp\_conf\_t, 406  
    vtss\_phy\_ts\_y1731\_oam\_conf\_t, 415  
lp\_advertisement  
    vtss\_eee\_port\_conf\_t, 90  
lp\_anegable  
    vtss\_phy\_10g\_kr\_status\_aneg\_t, 274  
lpcs\_1g  
    vtss\_phy\_10g\_status\_t, 322  
lref\_for\_host  
    vtss\_phys\_10g\_mode\_t, 290  
lrn\_all\_queue  
    vtss\_packet\_rx\_queue\_map\_t, 219  
ls\_inv  
    vtss\_phy\_ts\_sertod\_conf\_t, 411  
ls\_ipbk  
    vtss\_phy\_ts\_alt\_clock\_mode\_s, 365  
ls\_pps\_ipbk  
    vtss\_phy\_ts\_alt\_clock\_mode\_s, 365  
MAC\_ADDR\_BROADCAST  
    types.h, 610  
MAX\_CFG\_BUF\_SIZE  
    vtss\_phy\_api.h, 936  
MAX\_STAT\_BUF\_SIZE  
    vtss\_phy\_api.h, 936  
MAX\_WOL\_MAC\_ADDR\_SIZE  
    vtss\_phy\_api.h, 947  
MAX\_WOL\_PASSWD\_SIZE  
    vtss\_phy\_api.h, 947  
MIN\_WOL\_PASSWD\_SIZE  
    vtss\_phy\_api.h, 947  
mac  
    vtss\_ece\_key\_t, 82  
    vtss\_qce\_key\_t, 468  
    vtss\_vce\_key\_t, 530  
    vtss\_vid\_mac\_t, 537  
mac\_addr  
    vtss\_phy\_ts\_eth\_conf\_t, 373  
mac\_if  
    vtss\_phy\_reset\_conf\_t, 355  
mac\_if\_pcs  
    vtss\_phy\_conf\_t, 336  
mac\_serdes\_equipment\_enable  
    vtss\_phy\_loopback\_t, 344  
mac\_serdes\_facility\_enable  
    vtss\_phy\_loopback\_t, 343  
mac\_serdes\_input\_enable  
    vtss\_phy\_loopback\_t, 343  
mac\_vid\_queue  
    vtss\_packet\_rx\_queue\_map\_t, 218  
macsec\_ena  
    vtss\_phy\_ts\_init\_conf\_t, 389  
magic\_pkt\_cnt  
    vtss\_phy\_wol\_conf\_t, 420  
main\_status  
    vtss\_phy\_10g\_prbs\_mon\_conf\_t, 308  
map  
    vtss\_packet\_rx\_conf\_t, 199  
mask  
    vtss\_phy\_ts\_ach\_conf\_t, 363  
    vtss\_phy\_ts\_eth\_conf\_t, 375, 376  
    vtss\_phy\_ts\_gen\_conf\_t, 382  
    vtss\_phy\_ts\_generic\_action\_t, 384  
    vtss\_phy\_ts\_ip\_conf\_t, 393  
    vtss\_phy\_ts\_ptp\_conf\_t, 405  
    vtss\_phy\_ts\_y1731\_oam\_conf\_t, 415  
    vtss\_vcap\_ip\_t, 511  
    vtss\_vcap\_u128\_t, 512

vtss\_vcap\_u16\_t, 513  
 vtss\_vcap\_u24\_t, 514  
 vtss\_vcap\_u32\_t, 515  
 vtss\_vcap\_u40\_t, 516  
 vtss\_vcap\_u48\_t, 517  
 vtss\_vcap\_u8\_t, 518  
 vtss\_vcap\_vid\_t, 521  
 vtss\_vcap\_vr\_t, 522  
 masquerade\_port  
     vtss\_packet\_tx\_info\_t, 233  
 master  
     vtss\_phy\_10g\_mode\_t, 289  
     vtss\_phy\_conf\_1g\_t, 334  
     vtss\_phy\_status\_1g\_t, 360  
 master\_cfg\_fault  
     vtss\_phy\_status\_1g\_t, 360  
 match\_mode  
     vtss\_phy\_ts\_ip\_conf\_t, 393  
 max  
     ib\_par\_cfg, 32  
 max\_bist\_frames  
     vtss\_phy\_10g\_prbs\_mon\_conf\_t, 306  
 max\_frame\_length  
     vtss\_port\_conf\_t, 431  
 max\_length  
     port\_custom\_conf\_t, 34  
 max\_prob\_1  
     vtss\_red\_t, 487  
 max\_prob\_2  
     vtss\_red\_t, 487  
 max\_prob\_3  
     vtss\_red\_t, 487  
 max\_tags  
     port\_custom\_conf\_t, 35  
     vtss\_port\_conf\_t, 432  
 max\_th  
     vtss\_red\_t, 486  
 mc\_no\_flood  
     vtss\_policer\_ext\_t, 423  
 mdi  
     vtss\_phy\_conf\_t, 335  
 mdi\_cross  
     vtss\_port\_status\_t, 460  
 media\_if  
     vtss\_phy\_reset\_conf\_t, 355  
 media\_if\_pcs  
     vtss\_phy\_conf\_t, 336  
 media\_mac\_serdes\_crc  
     vtss\_phy\_statistic\_t, 359  
 media\_mac\_serdes\_good  
     vtss\_phy\_statistic\_t, 359  
 media\_serdes\_equipment\_enable  
     vtss\_phy\_loopback\_t, 344  
 media\_serdes\_facility\_enable  
     vtss\_phy\_loopback\_t, 344  
 media\_serdes\_input\_enable  
     vtss\_phy\_loopback\_t, 344  
 meg\_level  
     vtss\_phy\_ts\_y1731\_oam\_conf\_t, 415  
 miim\_addr  
     vtss\_port\_map\_t, 446  
 miim\_chip\_no  
     vtss\_port\_map\_t, 446  
 miim\_controller  
     vtss\_port\_map\_t, 446  
 miim\_read  
     vtss\_init\_conf\_t, 160  
 miim\_write  
     vtss\_init\_conf\_t, 161  
 min  
     ib\_par\_cfg, 31  
 min\_th  
     vtss\_red\_t, 486  
 mirror  
     vtss\_vlan\_vid\_conf\_t, 544  
     vtss\_vstax\_port\_conf\_t, 548  
 mld\_cpu\_only  
     vtss\_packet\_rx\_reg\_t, 222  
 mmd\_read  
     vtss\_init\_conf\_t, 161  
 mmd\_read\_inc  
     vtss\_init\_conf\_t, 161  
 mmd\_write  
     vtss\_init\_conf\_t, 161  
 mode  
     vtss\_ewis\_tti\_s, 132  
     vtss\_gpio\_10g\_gpio\_mode\_t, 156  
     vtss\_phy\_10g\_ckout\_conf\_t, 248  
     vtss\_phy\_10g\_host\_clk\_conf\_t, 260  
     vtss\_phy\_10g\_line\_clk\_conf\_t, 279  
     vtss\_phy\_10g\_rxckout\_conf\_t, 310  
     vtss\_phy\_10g\_sckout\_conf\_t, 311  
     vtss\_phy\_10g\_txckout\_conf\_t, 324  
     vtss\_phy\_conf\_t, 335  
     vtss\_phy\_led\_mode\_select\_t, 342  
     vtss\_phy\_power\_conf\_t, 353  
     vtss\_qs\_conf\_t, 483  
     vtss\_sgpi\_port\_conf\_t, 495  
     vtss\_ts\_operation\_mode\_t, 508  
 moved  
     vtss\_mac\_table\_status\_t, 188  
 mpls\_opt  
     vtss\_phy\_ts\_oam\_engine\_flow\_conf\_t, 403  
     vtss\_phy\_ts\_ptp\_engine\_flow\_conf\_t, 410  
 msg\_type  
     vtss\_phy\_ts\_fifo\_sig\_t, 379  
 multicast  
     vtss\_policer\_ext\_t, 422  
 mux\_mode  
     vtss\_init\_conf\_t, 163  
 n\_ebc\_thr\_l  
     vtss\_ewis\_counter\_threshold\_s, 107  
 n\_ebc\_thr\_p  
     vtss\_ewis\_counter\_threshold\_s, 108  
 n\_ebc\_thr\_s  
     vtss\_ewis\_counter\_threshold\_s, 107

nanoseconds  
  vtss\_phy\_timestamp\_t, 362  
  vtss\_timestamp\_t, 504

near\_end\_enable  
  vtss\_phy\_loopback\_t, 343

network  
  vtss\_evc\_conf\_t, 95  
  vtss\_ipv4\_uc\_t, 168  
  vtss\_ipv6\_uc\_t, 170

next  
  tag\_vtss\_fdma\_list, 39

next\_page  
  vtss\_phy\_10g\_clause\_37\_adv\_t, 251  
  vtss\_port\_clause\_37\_adv\_t, 428

next\_prot\_offset  
  vtss\_phy\_ts\_gen\_conf\_t, 381

nni  
  vtss\_evc\_pb\_conf\_t, 98

no\_of\_errors  
  vtss\_phy\_10g\_prbs\_mon\_conf\_t, 307

no\_sync  
  vtss\_phy\_10g\_prbs\_mon\_conf\_t, 309

no\_wait  
  vtss\_packet\_rx\_meta\_t, 212

now  
  vtss\_mtimer\_t, 191

npi  
  vtss\_packet\_rx\_queue\_conf\_t, 217

num\_tag  
  vtss\_phy\_ts\_eth\_conf\_t, 373

number  
  vtss\_phy\_led\_mode\_select\_t, 342

oam  
  vtss\_phy\_ts\_engine\_flow\_conf\_t, 370

oam\_conf  
  vtss\_phy\_ts\_engine\_action\_t, 368  
  vtss\_phy\_ts\_oam\_engine\_action\_t, 402

oam\_info  
  vtss\_packet\_rx\_info\_t, 209

oam\_info\_decoded  
  vtss\_packet\_rx\_info\_t, 210

oam\_type  
  vtss\_packet\_tx\_info\_t, 231

ob\_conf  
  vtss\_phy\_10g\_mode\_t, 285

ob\_post0  
  serdes\_fields\_t, 36

ob\_sr  
  serdes\_fields\_t, 36

obey  
  vtss\_port\_flow\_control\_conf\_t, 439

obey\_pause  
  vtss\_aneg\_t, 66

offs  
  vtss\_phy\_10g\_ib\_conf\_t, 262

one\_pps\_mode  
  vtss\_ts\_ext\_clock\_mode\_t, 505

one\_step\_txfifo

            vtss\_phy\_ts\_init\_conf\_t, 390

op\_enable  
  vtss\_phy\_ts\_sertod\_conf\_t, 411

op\_mode  
  vtss\_phy\_10g\_apc\_conf\_t, 234

op\_mode\_flag  
  vtss\_phy\_10g\_apc\_conf\_t, 235

oper\_mode  
  vtss\_phy\_10g\_mode\_t, 282

oper\_mode\_t  
  vtss\_phy\_10g\_api.h, 861

oper\_up  
  port\_custom\_conf\_t, 35

optimized\_for\_power  
  vtss\_eee\_port\_conf\_t, 90

options  
  vtss\_ace\_frame\_ipv4\_t, 45  
  vtss\_vce\_frame\_ipv4\_t, 526

options.h  
  VIPER\_B\_FIFO\_RESET, 574  
  VTSS\_ARCH\_JAGUAR\_1, 560  
  VTSS\_ARCH\_JAGUAR\_1\_CE\_SWITCH, 560  
  VTSS\_ARCH\_MALIBU\_B, 577  
  VTSS\_ARCH\_MALIBU, 577  
  VTSS\_ARCH\_VENICE\_C, 577  
  VTSS\_CHIP\_10G\_PHY, 562  
  VTSS\_CHIP CU PHY, 574  
  VTSS\_FEATURE\_10GBASE\_KR, 572  
  VTSS\_FEATURE\_10G, 563  
  VTSS\_FEATURE\_ACL\_V1, 571  
  VTSS\_FEATURE\_ACL, 571  
  VTSS\_FEATURE\_AFI\_FDMA, 573  
  VTSS\_FEATURE\_AGGR\_GLAG, 561  
  VTSS\_FEATURE\_CLAUSE\_37, 563  
  VTSS\_FEATURE\_E\_TREE, 560  
  VTSS\_FEATURE\_EDC\_FW\_LOAD, 576  
  VTSS\_FEATURE\_EEE, 572  
  VTSS\_FEATURE\_EVC, 560  
  VTSS\_FEATURE\_EXC\_COL\_CONT, 564  
  VTSS\_FEATURE\_FAN, 562, 572  
  VTSS\_FEATURE\_FDMA, 573  
  VTSS\_FEATURE\_IRQ\_CONTROL, 573  
  VTSS\_FEATURE\_LAYER2, 569  
  VTSS\_FEATURE\_LAYER3, 570  
  VTSS\_FEATURE\_LED\_POW\_REDUC, 572  
  VTSS\_FEATURE\_MAC\_AGE\_AUTO, 570  
  VTSS\_FEATURE\_MAC\_CPU\_QUEUE, 570  
  VTSS\_FEATURE\_MISC, 562  
  VTSS\_FEATURE\_NPI, 564  
  VTSS\_FEATURE\_PACKET\_GROUPING, 569  
  VTSS\_FEATURE\_PACKET\_PORT\_REG, 569  
  VTSS\_FEATURE\_PACKET\_RX, 568  
  VTSS\_FEATURE\_PACKET\_TX, 568  
  VTSS\_FEATURE\_PACKET, 568  
  VTSS\_FEATURE\_PHY\_TIMESTAMP, 561  
  VTSS\_FEATURE\_PORT\_CNT\_BRIDGE, 564  
  VTSS\_FEATURE\_PORT\_CNT\_ETHER\_LIKE, 564

VTSS FEATURE PORT CONTROL, [563](#)  
 VTSS FEATURE PORT IFH, [563](#)  
 VTSS FEATURE PORT MUX, [570](#)  
 VTSS FEATURE PVLAN, [562](#)  
 VTSS FEATURE QCL V2, [565](#)  
 VTSS FEATURE QCL, [565](#)  
 VTSS FEATURE QOS CLASSIFICATION V2, [567](#)  
 VTSS FEATURE QOS DSCP REMARK V2, [568](#)  
 VTSS FEATURE QOS DSCP REMARK, [567](#)  
 VTSS FEATURE QOS EGRESS QUEUE SH ← APERS EB, [567](#)  
 VTSS FEATURE QOS EGRESS QUEUE SH ← APERS, [567](#)  
 VTSS FEATURE QOS POLICER DLB, [574](#)  
 VTSS FEATURE QOS PORT POLICER EXT ← DPBL, [566](#)  
 VTSS FEATURE QOS PORT POLICER EXT ← FPS, [565](#)  
 VTSS FEATURE QOS PORT POLICER EXT ← FC, [565](#)  
 VTSS FEATURE QOS PORT POLICER EXT ← TTM, [566](#)  
 VTSS FEATURE QOS PORT POLICER EXT, [565](#)  
 VTSS FEATURE QOS QUEUE POLICER, [566](#)  
 VTSS FEATURE QOS QUEUE TX, [566](#)  
 VTSS FEATURE QOS SCHEDULER V2, [566](#)  
 VTSS FEATURE QOS TAG REMARK V2, [567](#)  
 VTSS FEATURE QOS WRED, [568](#)  
 VTSS FEATURE QOS, [564](#)  
 VTSS FEATURE SERDES MACRO SETTING GS, [572](#)  
 VTSS FEATURE SERIAL GPIO, [563](#)  
 VTSS FEATURE SFLOW, [574](#)  
 VTSS FEATURE SYNC 10G, [576](#)  
 VTSS FEATURE SYNC, [571](#)  
 VTSS FEATURE TIMESTAMP, [560](#)  
 VTSS FEATURE VCAP, [571](#), [577](#)  
 VTSS FEATURE VCL, [571](#)  
 VTSS FEATURE VLAN PORT V2, [569](#)  
 VTSS FEATURE VLAN SVL, [570](#)  
 VTSS FEATURE VLAN TRANSLATION, [573](#)  
 VTSS FEATURE VLAN TX TAG, [569](#)  
 VTSS FEATURE VSTAX V2, [561](#)  
 VTSS FEATURE VSTAX, [561](#)  
 VTSS FEATURE WARM START, [576](#)  
 VTSS FEATURE WIS, [576](#)  
 VTSS OPT FDMA DEBUG, [575](#)  
 VTSS OPT FDMA IRQ CONTEXT, [575](#)  
 VTSS OPT TRACE, [575](#)  
 VTSS OPT TS SPI FPGA, [562](#)  
 VTSS OPT VAUI EQ CTRL, [575](#)  
 VTSS OPT VCORE III, [573](#)  
 VTSS PHY 10G FIFO SYNC, [574](#)  
 VTSS PHY OPT VERIPH, [575](#)  
 VTSS PHY TS SPI CLK THRU PPS0, [561](#)

out\_of\_range  
 vtss\_rcpll\_status\_t, [485](#)  
 outer\_tag  
 vtss\_ece\_action\_t, [77](#)  
 vtss\_phy\_ts\_eth\_conf\_t, [375](#)  
 outer\_tag\_type  
 vtss\_phy\_ts\_eth\_conf\_t, [374](#)  
 oversubscription  
 vtss\_qs\_conf\_t, [483](#)  
 p\_gpio  
 vtss\_gpio\_10g\_gpio\_mode\_t, [157](#)  
 p\_gpio\_intrpt  
 vtss\_gpio\_10g\_gpio\_mode\_t, [158](#)  
**PHASE\_POINTS**  
 vtss\_phy\_10g\_api.h, [848](#)  
**PORT\_CAP\_100M\_FDX**  
 port.h, [583](#)  
**PORT\_CAP\_100M\_HDX**  
 port.h, [582](#)  
**PORT\_CAP\_10G\_FDX**  
 port.h, [583](#)  
**PORT\_CAP\_10M\_FDX**  
 port.h, [582](#)  
**PORT\_CAP\_10M\_HDX**  
 port.h, [582](#)  
**PORT\_CAP\_1G\_FDX**  
 port.h, [583](#)  
**PORT\_CAP\_1G\_PHY**  
 port.h, [587](#)  
**PORT\_CAP\_2\_5G\_FDX**  
 port.h, [583](#)  
**PORT\_CAP\_2\_5G\_TRI\_SPEED\_COPPER**  
 port.h, [591](#)  
**PORT\_CAP\_2\_5G\_TRI\_SPEED\_FDX**  
 port.h, [590](#)  
**PORT\_CAP\_2\_5G\_TRI\_SPEED**  
 port.h, [591](#)  
**PORT\_CAP\_5G\_FDX**  
 port.h, [583](#)  
**PORT\_CAP\_ANY\_FIBER**  
 port.h, [589](#)  
**PORT\_CAP\_AUTONEG**  
 port.h, [582](#)  
**PORT\_CAP\_COPPER**  
 port.h, [584](#)  
**PORT\_CAP\_DUAL\_COPPER\_100FX**  
 port.h, [587](#)  
**PORT\_CAP\_DUAL\_COPPER**  
 port.h, [584](#)  
**PORT\_CAP\_DUAL\_FIBER\_1000X**  
 port.h, [590](#)  
**PORT\_CAP\_DUAL\_FIBER\_100FX**  
 port.h, [585](#)  
**PORT\_CAP\_DUAL\_FIBER**  
 port.h, [584](#)  
**PORT\_CAP\_DUAL\_SFP\_DETECT**  
 port.h, [586](#)  
**PORT\_CAP\_FIBER**

port.h, 584  
PORT\_CAP\_FLOW\_CTRL  
    port.h, 584  
PORT\_CAP\_HDX  
    port.h, 587  
PORT\_CAP\_NONE  
    port.h, 582  
PORT\_CAP\_SD\_ENABLE  
    port.h, 585  
PORT\_CAP\_SD\_HIGH  
    port.h, 585  
PORT\_CAP\_SD\_INTERNAL  
    port.h, 585  
PORT\_CAP\_SFP\_1G  
    port.h, 590  
PORT\_CAP\_SFP\_2\_5G  
    port.h, 590  
PORT\_CAP\_SFP\_DETECT  
    port.h, 586  
PORT\_CAP\_SFP\_ONLY  
    port.h, 586  
PORT\_CAP\_SFP\_SD\_HIGH  
    port.h, 590  
PORT\_CAP\_SPEED\_DUAL\_ANY\_FIBER\_FIXED\_S←  
    PEED  
    port.h, 589  
PORT\_CAP\_SPEED\_DUAL\_ANY\_FIBER  
    port.h, 589  
PORT\_CAP\_STACKING  
    port.h, 586  
PORT\_CAP\_TRI\_SPEED\_COPPER  
    port.h, 588  
PORT\_CAP\_TRI\_SPEED\_DUAL\_ANY\_FIBER\_FIXE←  
    D\_SFP\_SPEED  
    port.h, 589  
PORT\_CAP\_TRI\_SPEED\_DUAL\_ANY\_FIBER  
    port.h, 589  
PORT\_CAP\_TRI\_SPEED\_DUAL\_COPPER  
    port.h, 588  
PORT\_CAP\_TRI\_SPEED\_DUAL\_FIBER\_100FX  
    port.h, 588  
PORT\_CAP\_TRI\_SPEED\_DUAL\_FIBER  
    port.h, 588  
PORT\_CAP\_TRI\_SPEED\_FDX  
    port.h, 587  
PORT\_CAP\_TRI\_SPEED\_FIBER  
    port.h, 588  
PORT\_CAP\_TRI\_SPEED  
    port.h, 587  
PORT\_CAP\_VTSS\_10G\_PHY  
    port.h, 586  
PORT\_CAP\_XAUI\_LANE\_FLIP  
    port.h, 585  
PRBS\_status  
    vtss\_phy\_10g\_prbs\_mon\_conf\_t, 308  
PRI64  
    types.h, 600  
PRIu64  
    types.h, 600  
PRIx64  
    types.h, 600  
part\_number  
    vtss\_chip\_id\_t, 69  
    vtss\_phy\_10g\_id\_t, 268  
    vtss\_phy\_type\_t, 416  
partner\_advertisement  
    vtss\_phy\_10g\_clause\_37\_status\_t, 255  
passwd  
    vtss\_secure\_on\_passwd\_t, 491  
path\_force  
    vtss\_ewis\_force\_mode\_s, 116  
path\_txti  
    vtss\_ewis\_conf\_s, 103  
pb  
    vtss\_evc\_conf\_t, 95  
pbb\_en  
    vtss\_phy\_ts\_eth\_conf\_t, 372  
pcp  
    vtss\_ece\_inner\_tag\_t, 81  
    vtss\_ece\_outer\_tag\_t, 86  
    vtss\_ece\_tag\_t, 88  
    vtss\_mirror\_conf\_t, 189  
    vtss\_qce\_tag\_t, 473  
    vtss\_vce\_tag\_t, 535  
    vtss\_vlan\_tag\_t, 541  
pcp\_dei\_preserve  
    vtss\_ece\_inner\_tag\_t, 81  
    vtss\_ece\_outer\_tag\_t, 85  
pcs  
    vtss\_phy\_10g\_cnt\_t, 256  
    vtss\_phy\_10g\_status\_t, 321  
pd\_enable  
    vtss\_phy\_mac\_serdes\_pcs\_ctrl\_t, 347  
pdu\_offset  
    vtss\_packet\_tx\_info\_t, 233  
perf\_mode  
    vtss\_ewis\_conf\_s, 104  
pf\_ebc\_l  
    vtss\_ewis\_counter\_s, 106  
    vtss\_ewis\_perf\_s, 121  
pf\_ebc\_mode\_l  
    vtss\_ewis\_perf\_mode\_s, 120  
pf\_ebc\_mode\_p  
    vtss\_ewis\_perf\_mode\_s, 120  
pf\_ebc\_p  
    vtss\_ewis\_counter\_s, 106  
    vtss\_ewis\_perf\_s, 122  
pfc  
    port\_custom\_conf\_t, 33  
    vtss\_port\_flow\_control\_conf\_t, 439  
phy.h  
    VTSS\_PHY\_POWER\_ACTIPHYS\_BIT, 578  
    VTSS\_PHY\_POWER\_DYNAMIC\_BIT, 578  
    vtss\_phy\_power\_mode\_t, 579  
    vtss\_phy\_veriphy\_status\_t, 579  
phy\_api\_base\_no

vtss\_phy\_10g\_id\_t, 269  
 vtss\_phy\_type\_t, 417  
 pi  
     vtss\_init\_conf\_t, 163  
 pkt\_len  
     vtss\_phy\_10g\_pkt\_gen\_conf\_t, 299  
 pkt\_mode  
     vtss\_phy\_reset\_conf\_t, 356  
 pma  
     vtss\_phy\_10g\_status\_t, 321  
 pma\_txratecontrol  
     vtss\_phy\_10g\_mode\_t, 288  
 pn\_ebc\_l  
     vtss\_ewis\_counter\_s, 106  
     vtss\_ewis\_perf\_s, 121  
 pn\_ebc\_mode\_l  
     vtss\_ewis\_perf\_mode\_s, 120  
 pn\_ebc\_mode\_p  
     vtss\_ewis\_perf\_mode\_s, 120  
 pn\_ebc\_mode\_s  
     vtss\_ewis\_perf\_mode\_s, 119  
 pn\_ebc\_p  
     vtss\_ewis\_counter\_s, 106  
     vtss\_ewis\_perf\_s, 122  
 pn\_ebc\_s  
     vtss\_ewis\_counter\_s, 106  
     vtss\_ewis\_perf\_s, 121  
 polarity  
     vtss\_phy\_10g\_mode\_t, 289  
 police  
     vtss\_acl\_action\_t, 61  
 policer\_ext\_port  
     vtss\_qos\_port\_conf\_t, 478  
 policer\_id  
     vtss\_ece\_action\_t, 78  
     vtss\_evc\_conf\_t, 94  
 policer\_no  
     vtss\_acl\_action\_t, 61  
 policer\_port  
     vtss\_qos\_port\_conf\_t, 477  
 policer\_queue  
     vtss\_qos\_port\_conf\_t, 478  
 policy  
     vtss\_ace\_t, 56  
 policy\_no  
     vtss\_acl\_port\_conf\_t, 64  
     vtss\_ece\_action\_t, 78  
     vtss\_vce\_action\_t, 524  
 pop\_tag  
     vtss\_ece\_action\_t, 77  
 port  
     vtss\_gpio\_10g\_gpio\_mode\_t, 156  
     vtss\_qs\_conf\_t, 484  
 port.h  
     PORT\_CAP\_100M\_FDX, 583  
     PORT\_CAP\_100M\_HDX, 582  
     PORT\_CAP\_10G\_FDX, 583  
     PORT\_CAP\_10M\_FDX, 582  
     PORT\_CAP\_10M\_HDX, 582  
     PORT\_CAP\_1G\_FDX, 583  
     PORT\_CAP\_1G\_PHY, 587  
     PORT\_CAP\_2\_5G\_FDX, 583  
     PORT\_CAP\_2\_5G\_TRI\_SPEED\_COPPER, 591  
     PORT\_CAP\_2\_5G\_TRI\_SPEED\_FDX, 590  
     PORT\_CAP\_2\_5G\_TRI\_SPEED, 591  
     PORT\_CAP\_5G\_FDX, 583  
     PORT\_CAP\_ANY\_FIBER, 589  
     PORT\_CAP\_AUTONEG, 582  
     PORT\_CAP\_COPPER, 584  
     PORT\_CAP\_DUAL\_COPPER\_100FX, 587  
     PORT\_CAP\_DUAL\_COPPER, 584  
     PORT\_CAP\_DUAL\_FIBER\_1000X, 590  
     PORT\_CAP\_DUAL\_FIBER\_100FX, 585  
     PORT\_CAP\_DUAL\_FIBER, 584  
     PORT\_CAP\_DUAL\_SFP\_DETECT, 586  
     PORT\_CAP\_FIBER, 584  
     PORT\_CAP\_FLOW\_CTRL, 584  
     PORT\_CAP\_HDX, 587  
     PORT\_CAP\_NONE, 582  
     PORT\_CAP\_SD\_ENABLE, 585  
     PORT\_CAP\_SD\_HIGH, 585  
     PORT\_CAP\_SD\_INTERNAL, 585  
     PORT\_CAP\_SFP\_1G, 590  
     PORT\_CAP\_SFP\_2\_5G, 590  
     PORT\_CAP\_SFP\_DETECT, 586  
     PORT\_CAP\_SFP\_ONLY, 586  
     PORT\_CAP\_SFP\_SD\_HIGH, 590  
     PORT\_CAP\_SPEED\_DUAL\_ANY\_FIBER\_FIXE←  
         D\_SPEED, 589  
     PORT\_CAP\_SPEED\_DUAL\_ANY\_FIBER, 589  
     PORT\_CAP\_STACKING, 586  
     PORT\_CAP\_TRI\_SPEED\_COPPER, 588  
     PORT\_CAP\_TRI\_SPEED\_DUAL\_ANY\_FIBER←  
         FIXED\_SFP\_SPEED, 589  
     PORT\_CAP\_TRI\_SPEED\_DUAL\_ANY\_FIBER,  
         589  
     PORT\_CAP\_TRI\_SPEED\_DUAL\_COPPER, 588  
     PORT\_CAP\_TRI\_SPEED\_DUAL\_FIBER\_100FX,  
         588  
     PORT\_CAP\_TRI\_SPEED\_DUAL\_FIBER, 588  
     PORT\_CAP\_TRI\_SPEED\_FDX, 587  
     PORT\_CAP\_TRI\_SPEED\_FIBER, 588  
     PORT\_CAP\_TRI\_SPEED, 587  
     PORT\_CAP\_VTSS\_10G\_PHY, 586  
     PORT\_CAP\_XAUI\_LANE\_FLIP, 585  
     port\_cap\_t, 591  
     vtss\_fiber\_port\_speed\_t, 592  
     vtss\_port\_speed\_t, 591  
 port\_0  
     vtss\_vstax\_conf\_t, 546  
 port\_1  
     vtss\_vstax\_conf\_t, 546  
 port\_cap\_t  
     port.h, 591  
 port\_cnt  
     vtss\_phy\_type\_t, 417

port\_conf  
    vtss\_sgpio\_conf\_t, 494

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, 35  
    oper\_up, 35  
    pfc, 33  
    power\_mode, 34  
    speed, 33

port\_forward  
    vtss\_acl\_action\_t, 62

port\_list  
    vtss\_debug\_info\_t, 72  
    vtss\_ece\_key\_t, 82  
    vtss\_qce\_key\_t, 468  
    vtss\_vce\_key\_t, 530

port\_mask  
    vtss\_ts\_timestamp\_alloc\_t, 509

port\_max  
    vtss\_qs\_conf\_t, 483

port\_min  
    vtss\_qs\_conf\_t, 483

port\_no  
    vtss\_ace\_t, 56  
    vtss\_acl\_action\_t, 62  
    vtss\_eps\_port\_conf\_t, 94  
    vtss\_mirror\_conf\_t, 189  
    vtss\_npi\_conf\_t, 192  
    vtss\_packet\_frame\_info\_t, 194  
    vtss\_packet\_port\_info\_t, 197  
    vtss\_packet\_rx\_header\_t, 200  
    vtss\_packet\_rx\_info\_t, 203  
    vtss\_phy\_10g\_auto\_failover\_conf\_t, 237  
    vtss\_sync\_clock\_in\_t, 499  
    vtss\_vstax\_rx\_header\_t, 551  
    vtss\_vstax\_tx\_header\_t, 553

port\_tx  
    vtss\_packet\_frame\_info\_t, 195

port\_type  
    vtss\_vlan\_port\_conf\_t, 539

ports  
    vtss\_vlan\_trans\_port2grp\_conf\_t, 543

power\_down  
    vtss\_port\_conf\_t, 431

power\_mode  
    port\_custom\_conf\_t, 34

ppr  
    vtss\_fan\_conf\_t, 141

pps\_ls\_lpbk  
    vtss\_phy\_ts\_alt\_clock\_mode\_s, 365

pps\_offset  
    vtss\_phy\_ts\_pps\_config\_s, 404

pps\_output\_enable  
    vtss\_phy\_ts\_pps\_config\_s, 404

pps\_width\_adj  
    vtss\_phy\_ts\_pps\_config\_s, 404

pr  
    vtss\_phy\_10g\_fifo\_sync\_t, 257

prbs  
    vtss\_phy\_10g\_ib\_conf\_t, 264

prbs\_check\_input\_invert  
    vtss\_phy\_10g\_prbs\_mon\_conf\_t, 307

prbs\_gen  
    vtss\_phy\_10g\_pcs\_prbs\_gen\_conf\_t, 296

prbs\_inv  
    vtss\_phy\_10g\_ib\_conf\_t, 264

prbs\_mon  
    vtss\_phy\_10g\_pcs\_prbs\_mon\_conf\_t, 296

prbsn\_sel  
    vtss\_phy\_10g\_prbs\_mon\_conf\_t, 307

prbsn\_tx\_io  
    vtss\_phy\_10g\_prbs\_gen\_conf\_t, 305

prbsn\_tx\_iw  
    vtss\_phy\_10g\_prbs\_gen\_conf\_t, 305

prbsn\_tx\_sel  
    vtss\_phy\_10g\_prbs\_gen\_conf\_t, 304

pre\_cb  
    vtss\_fdma\_tx\_info\_t, 153

pre\_cb\_ctxt1  
    vtss\_fdma\_tx\_info\_t, 153

pre\_cb\_ctxt2  
    vtss\_fdma\_tx\_info\_t, 153

prefix\_size  
    vtss\_ip\_network\_t, 166  
    vtss\_ipv4\_network\_t, 167  
    vtss\_ipv6\_network\_t, 169

prescale  
    vtss\_phy\_10g\_i2c\_slave\_conf\_t, 261

prev\_version  
    vtss\_restart\_status\_t, 488

prio  
    vtss\_fdma\_ch\_cfg\_t, 149  
    vtss\_qce\_action\_t, 461  
    vtss\_vstax\_tx\_header\_t, 553

prio\_enable  
    vtss\_qce\_action\_t, 461

prios  
    vtss\_qos\_conf\_t, 475

prop  
    vtss\_port\_counters\_t, 434

proto  
    vtss\_ace\_frame\_ipv4\_t, 45  
    vtss\_ace\_frame\_ipv6\_t, 49  
    vtss\_qce\_frame\_ipv4\_t, 464  
    vtss\_qce\_frame\_ipv6\_t, 465  
    vtss\_vce\_frame\_ipv4\_t, 526  
    vtss\_vce\_frame\_ipv6\_t, 527

proto\_id

vtss\_phy\_ts\_ach\_conf\_t, 364  
 ptp  
   vtss\_phy\_10g\_pkt\_gen\_conf\_t, 298  
   vtss\_phy\_ts\_engine\_flow\_conf\_t, 370  
 ptp\_action  
   vtss\_packet\_tx\_info\_t, 229  
 ptp\_conf  
   vtss\_phy\_ts\_engine\_action\_t, 368  
   vtss\_phy\_ts\_ptp\_engine\_action\_t, 407  
 ptp\_id  
   vtss\_packet\_tx\_info\_t, 229  
 ptp\_timestamp  
   vtss\_packet\_tx\_info\_t, 230  
 ptp\_ts\_ns  
   vtss\_phy\_10g\_pkt\_gen\_conf\_t, 300  
 ptp\_ts\_sec  
   vtss\_phy\_10g\_pkt\_gen\_conf\_t, 299  
 pvid  
   vtss\_vlan\_port\_conf\_t, 539  
 qos\_class\_map  
   vtss\_qos\_port\_conf\_t, 480  
 qs\_conf  
   vtss\_init\_conf\_t, 163  
 qsgmii  
   vtss\_serdes\_macro\_conf\_t, 492  
 queue  
   vtss\_packet\_rx\_conf\_t, 198  
 queue\_mask  
   vtss\_packet\_rx\_header\_t, 200  
 queue\_max  
   vtss\_qs\_conf\_t, 484  
 queue\_min  
   vtss\_qs\_conf\_t, 484  
 queue\_no  
   vtss\_vstax\_tx\_header\_t, 554  
 queue\_pct  
   vtss\_qos\_port\_conf\_t, 482  
  
 r  
   vtss\_vcap\_vr\_t, 523  
 r\_ctrl  
   vtss\_phy\_10g\_ob\_status\_t, 293  
 range  
   vtss\_phy\_ts\_eth\_conf\_t, 375  
   vtss\_phy\_ts\_ptp\_conf\_t, 406  
   vtss\_phy\_ts\_y1731\_oam\_conf\_t, 415  
 range\_en  
   vtss\_phy\_ts\_ptp\_conf\_t, 405  
   vtss\_phy\_ts\_y1731\_oam\_conf\_t, 414  
 rate  
   vtss\_acl\_policer\_conf\_t, 63  
   vtss\_phy\_10g\_mode\_t, 289  
   vtss\_policer\_t, 425  
   vtss\_shaper\_t, 497  
 raw\_ident  
   vtss\_irq\_status\_t, 172  
 raw\_mask  
   vtss\_irq\_status\_t, 173  
  
 raw\_status  
   vtss\_irq\_status\_t, 173  
 rcomp  
   vtss\_phy\_10g\_serd़es\_status\_t, 313  
 rcvrd\_clk  
   vtss\_phy\_10g\_mode\_t, 284  
 rcvrd\_clk\_div  
   vtss\_phy\_10g\_mode\_t, 284  
 rdil  
   vtss\_ewis\_cons\_act\_s, 105  
 rdil\_on\_ais\_l  
   vtss\_ewis\_rdil\_cons\_act\_s, 123  
 rdil\_on\_lof  
   vtss\_ewis\_rdil\_cons\_act\_s, 123  
 rdil\_on\_lopc  
   vtss\_ewis\_rdil\_cons\_act\_s, 123  
 rdil\_on\_los  
   vtss\_ewis\_rdil\_cons\_act\_s, 123  
 recvrd\_clk\_sel  
   vtss\_phy\_10g\_host\_clk\_conf\_t, 260  
   vtss\_phy\_10g\_line\_clk\_conf\_t, 279  
 red  
   vtss\_qos\_port\_conf\_t, 477  
 reg  
   vtss\_packet\_rx\_conf\_t, 199  
 reg\_read  
   vtss\_init\_conf\_t, 160  
 reg\_write  
   vtss\_init\_conf\_t, 160  
 remote\_entry  
   vtss\_mac\_table\_entry\_t, 186  
 remote\_fault  
   vtss\_phy\_10g\_clause\_37\_adv\_t, 250  
   vtss\_phy\_media\_serd़es\_pcs\_cntl\_t, 350  
   vtss\_port\_clause\_37\_adv\_t, 427  
   vtss\_port\_status\_t, 459  
 replaced  
   vtss\_mac\_table\_status\_t, 187  
 req  
   vtss\_ace\_frame\_arp\_t, 40  
 request\_10g  
   vtss\_phy\_10g\_kr\_status\_aneg\_t, 273  
 request\_1g  
   vtss\_phy\_10g\_kr\_status\_aneg\_t, 273  
 request\_fec\_change  
   vtss\_phy\_10g\_kr\_status\_aneg\_t, 273  
 reset  
   vtss\_phy\_10g\_apc\_status\_t, 235  
   vtss\_phy\_10g\_pkt\_mon\_conf\_t, 301  
 restart  
   vtss\_phy\_mac\_serd़es\_pcs\_cntl\_t, 347  
   vtss\_restart\_status\_t, 488  
 restart\_info\_port  
   vtss\_init\_conf\_t, 162  
 restart\_info\_src  
   vtss\_init\_conf\_t, 162  
 revision  
   vtss\_chip\_id\_t, 69

vtss\_phy\_10g\_id\_t, 268  
vtss\_phy\_type\_t, 416  
rgmii  
    vtss\_phy\_reset\_conf\_t, 355  
rgmii\_skew\_delay\_psec\_t  
    vtss\_phy\_api.h, 950  
rleg\_mode  
    vtss\_l3\_common\_conf\_t, 174  
rmon  
    vtss\_port\_counters\_t, 434  
route  
    vtss\_routing\_entry\_t, 490  
routing\_enable  
    vtss\_l3\_common\_conf\_t, 174  
rx\_alloc\_cb  
    vtss\_fdma\_cfg\_t, 143  
rx\_allow\_multiple\_dcbs  
    vtss\_fdma\_cfg\_t, 145  
rx\_allow\_vlan\_tag\_mismatch  
    vtss\_fdma\_cfg\_t, 144  
rx\_buf\_cnt  
    vtss\_fdma\_cfg\_t, 143  
rx\_cb  
    vtss\_fdma\_cfg\_t, 145  
rx\_ch  
    vtss\_phy\_10g\_lane\_sync\_conf\_t, 278  
rx\_clk\_skew\_ps  
    vtss\_phy\_rgmii\_conf\_t, 357  
rx\_discard  
    vtss\_evc\_counters\_t, 96  
rx\_dont\_reinsert\_vlan\_tag  
    vtss\_fdma\_cfg\_t, 144  
rx\_dont\_strip\_vlan\_tag  
    vtss\_fdma\_cfg\_t, 144  
rx\_err\_cnt\_base\_tx  
    vtss\_phy\_statistic\_t, 358  
rx\_etherStatsBroadcastPkts  
    vtss\_port\_rmon\_counters\_t, 449  
rx\_etherStatsCRCAlignErrors  
    vtss\_port\_rmon\_counters\_t, 450  
rx\_etherStatsDropEvents  
    vtss\_port\_rmon\_counters\_t, 449  
rx\_etherStatsFragments  
    vtss\_port\_rmon\_counters\_t, 450  
rx\_etherStatsJabbers  
    vtss\_port\_rmon\_counters\_t, 450  
rx\_etherStatsMulticastPkts  
    vtss\_port\_rmon\_counters\_t, 449  
rx\_etherStatsOctets  
    vtss\_port\_rmon\_counters\_t, 449  
rx\_etherStatsOversizePkts  
    vtss\_port\_rmon\_counters\_t, 450  
rx\_etherStatsPkts  
    vtss\_port\_rmon\_counters\_t, 449  
rx\_etherStatsPkts1024to1518Octets  
    vtss\_port\_rmon\_counters\_t, 452  
rx\_etherStatsPkts128to255Octets  
    vtss\_port\_rmon\_counters\_t, 451  
rx\_etherStatsPkts1519toMaxOctets  
    vtss\_port\_rmon\_counters\_t, 452  
rx\_etherStatsPkts256to511Octets  
    vtss\_port\_rmon\_counters\_t, 451  
rx\_etherStatsPkts512to1023Octets  
    vtss\_port\_rmon\_counters\_t, 451  
rx\_etherStatsPkts64Octets  
    vtss\_port\_rmon\_counters\_t, 451  
rx\_etherStatsPkts65to127Octets  
    vtss\_port\_rmon\_counters\_t, 451  
rx\_etherStatsUndersizePkts  
    vtss\_port\_rmon\_counters\_t, 450  
rx\_fault  
    vtss\_sublayer\_status\_t, 498  
rx\_frames  
    vtss\_basic\_counters\_t, 68  
rx\_green  
    vtss\_evc\_counters\_t, 96  
rx\_info  
    tag\_vtss\_fdma\_list, 38  
rx\_link  
    vtss\_sublayer\_status\_t, 498  
rx\_macro  
    vtss\_phy\_10g\_lane\_sync\_conf\_t, 277  
rx\_mtu  
    vtss\_fdma\_cfg\_t, 142  
rx\_prio  
    vtss\_port\_proprietary\_counters\_t, 447  
rx\_red  
    vtss\_evc\_counters\_t, 96  
rx\_ts\_len  
    vtss\_phy\_ts\_init\_conf\_t, 388  
rx\_ts\_pos  
    vtss\_phy\_ts\_init\_conf\_t, 387  
rx\_yellow  
    vtss\_evc\_counters\_t, 96  
sETYPE  
    vtss\_vlan\_conf\_t, 538  
s\_TAG  
    vtss\_vce\_tag\_t, 536  
s\_TAGGED  
    vtss\_ece\_tag\_t, 88  
sampling\_rate  
    vtss\_sflow\_port\_conf\_t, 493  
scan\_CONF  
    vtss\_phy\_10g\_vsphere\_scan\_status\_t, 328  
scan\_TYPE  
    vtss\_phy\_10g\_vsphere\_conf\_t, 325  
sd6g\_calib\_done  
    vtss\_phy\_10g\_mode\_t, 292  
sd\_ACTIVE\_HIGH  
    vtss\_port\_conf\_t, 430  
sd\_ENABLE  
    vtss\_port\_conf\_t, 430  
sd\_INTERNAL  
    vtss\_port\_conf\_t, 430  
sec  
    vtss\_timeofday\_t, 502, 503

sec\_msbs  
     vtss\_timestamp\_t, 503  
 seconds  
     vtss\_phy\_timestamp\_t, 362  
     vtss\_timestamp\_t, 504  
 section\_cons\_act  
     vtss\_ewis\_conf\_s, 102  
 section\_txiti  
     vtss\_ewis\_conf\_s, 102  
 secure\_on\_enable  
     vtss\_phy\_wol\_conf\_t, 419  
 select  
     vtss\_eee\_port\_state\_t, 92  
 seq\_zero  
     vtss\_ace\_frame\_ipv4\_t, 48  
     vtss\_ace\_frame\_ipv6\_t, 52  
 sequence\_id  
     vtss\_phy\_ts\_fifo\_sig\_t, 379  
 ser\_inv  
     vtss\_phy\_10g\_base\_kr\_conf\_t, 241  
 ser\_led\_frame\_rate  
     vtss\_phy\_enhanced\_led\_control\_t, 340  
 ser\_led\_output\_1  
     vtss\_phy\_enhanced\_led\_control\_t, 339  
 ser\_led\_output\_2  
     vtss\_phy\_enhanced\_led\_control\_t, 340  
 ser\_led\_select  
     vtss\_phy\_enhanced\_led\_control\_t, 340  
 serdes  
     vtss\_init\_conf\_t, 163  
     vtss\_port\_conf\_t, 433  
 serdes1g\_vdd  
     vtss\_serdes\_macro\_conf\_t, 491  
 serdes6g\_vdd  
     vtss\_serdes\_macro\_conf\_t, 491  
 serdes\_aneg\_ena  
     vtss\_phy\_mac\_serdes\_pcs\_ctrl\_t, 348  
 serdes\_conf  
     vtss\_phy\_10g\_mode\_t, 288  
 serdes\_fields\_t, 35  
     ob\_post0, 36  
     ob\_sr, 36  
 serdes\_pol\_inv\_in  
     vtss\_phy\_mac\_serdes\_pcs\_ctrl\_t, 348  
     vtss\_phy\_media\_serdes\_pcs\_ctrl\_t, 350  
 serdes\_pol\_inv\_out  
     vtss\_phy\_mac\_serdes\_pcs\_ctrl\_t, 348  
     vtss\_phy\_media\_serdes\_pcs\_ctrl\_t, 350  
 serdes\_tx\_bad  
     vtss\_phy\_statistic\_t, 358  
 serdes\_tx\_good  
     vtss\_phy\_statistic\_t, 358  
 sflow\_port\_no  
     vtss\_packet\_rx\_info\_t, 209  
 sflow\_queue  
     vtss\_packet\_rx\_queue\_map\_t, 219  
 sflow\_type  
     vtss\_packet\_rx\_info\_t, 208  
 sfp\_dac  
     vtss\_port\_serdes\_conf\_t, 455  
 sgmii\_in\_pre  
     vtss\_phy\_mac\_serdes\_pcs\_ctrl\_t, 347  
 sgmii\_out\_pre  
     vtss\_phy\_mac\_serdes\_pcs\_ctrl\_t, 348  
 shaper\_port  
     vtss\_qos\_port\_conf\_t, 478  
 shaper\_queue  
     vtss\_qos\_port\_conf\_t, 478  
 sig\_det  
     vtss\_phy\_10g\_ib\_status\_t, 266  
 sig\_mask  
     vtss\_phy\_ts\_fifo\_sig\_t, 379  
 sigdet  
     vtss\_phy\_conf\_t, 336  
 sip  
     vtss\_ace\_frame\_arp\_t, 42  
     vtss\_ace\_frame\_ipv4\_t, 45  
     vtss\_ace\_frame\_ipv6\_t, 49  
     vtss\_qce\_frame\_ipv4\_t, 464  
     vtss\_qce\_frame\_ipv6\_t, 466  
     vtss\_vce\_frame\_ipv4\_t, 526  
     vtss\_vce\_frame\_ipv6\_t, 528  
 sip\_dip\_enable  
     vtss\_aggr\_mode\_t, 65  
 sip\_eq\_dip  
     vtss\_ace\_frame\_ipv4\_t, 48  
     vtss\_ace\_frame\_ipv6\_t, 52  
 size  
     vtss\_packet\_rx\_queue\_conf\_t, 216  
 skip\_coma  
     vtss\_phy\_conf\_t, 337  
 skip\_rev\_check  
     vtss\_phy\_10g\_fifo\_sync\_t, 257  
     vtss\_phy\_ts\_fifo\_conf\_t, 378  
 slave\_id  
     vtss\_phy\_10g\_i2c\_slave\_conf\_t, 261  
 slew  
     vtss\_phy\_10g\_ob\_status\_t, 293  
 slewrate  
     vtss\_phy\_10g\_base\_kr\_conf\_t, 241  
 sm  
     vtss\_phy\_10g\_kr\_status\_aneg\_t, 273  
 smac  
     vtss\_ace\_frame\_arp\_t, 40  
     vtss\_ace\_frame\_etype\_t, 43  
     vtss\_ace\_frame\_llc\_t, 53  
     vtss\_ace\_frame\_snap\_t, 54  
     vtss\_phy\_10g\_pkt\_gen\_conf\_t, 299  
     vtss\_port\_flow\_control\_conf\_t, 439  
     vtss\_qce\_mac\_t, 471  
     vtss\_vce\_mac\_t, 533  
 smac\_enable  
     vtss\_aggr\_mode\_t, 65  
 smac\_match  
     vtss\_ace\_frame\_arp\_t, 41  
 snap

vtss\_ace\_frame\_snap\_t, 54  
vtss\_ace\_t, 57  
vtss\_qce\_key\_t, 469  
vtss\_vce\_key\_t, 531  
snd\_lvl\_after\_top  
    vtss\_phy\_ts\_mpls\_conf\_t, 396  
snd\_lvl\_before\_end  
    vtss\_phy\_ts\_mpls\_conf\_t, 397  
source  
    vtss\_gpio\_10g\_gpio\_mode\_t, 157  
sp  
    vtss\_vstax\_rx\_header\_t, 551  
speed  
    port\_custom\_conf\_t, 33  
    vtss\_phy\_forced\_t, 341  
    vtss\_port\_conf\_t, 431  
    vtss\_port\_status\_t, 458  
speed\_100M  
    vtss\_port\_sgmii\_aneg\_t, 457  
speed\_100m\_fdx  
    vtss\_phy\_aneg\_t, 331  
speed\_100m\_hdx  
    vtss\_phy\_aneg\_t, 330  
speed\_10M  
    vtss\_port\_sgmii\_aneg\_t, 457  
speed\_10m\_fdx  
    vtss\_phy\_aneg\_t, 330  
speed\_10m\_hdx  
    vtss\_phy\_aneg\_t, 330  
speed\_1G  
    vtss\_port\_sgmii\_aneg\_t, 457  
speed\_1g\_fdx  
    vtss\_phy\_aneg\_t, 331  
speed\_1g\_hdx  
    vtss\_phy\_aneg\_t, 331  
spi\_32bit\_read\_write  
    vtss\_init\_conf\_t, 162  
spi\_64bit\_read\_write  
    vtss\_init\_conf\_t, 162  
spi\_daisy\_input  
    vtss\_phy\_daisy\_chain\_conf\_t, 338  
spi\_daisy\_output  
    vtss\_phy\_daisy\_chain\_conf\_t, 338  
spi\_read\_write  
    vtss\_init\_conf\_t, 161  
sport  
    vtss\_ace\_frame\_ipv4\_t, 46  
    vtss\_ace\_frame\_ipv6\_t, 50  
    vtss\_qce\_frame\_ipv4\_t, 464  
    vtss\_qce\_frame\_ipv6\_t, 466  
sport\_dport\_enable  
    vtss\_aggr\_mode\_t, 65  
sport\_eq\_dport  
    vtss\_ace\_frame\_ipv4\_t, 48  
    vtss\_ace\_frame\_ipv6\_t, 52  
sport\_mask  
    vtss\_phy\_ts\_ip\_conf\_t, 392  
sport\_val

vtss\_phy\_ts\_ip\_conf\_t, 391  
squelch  
    vtss\_phy\_clock\_conf\_t, 333  
squelch\_inv  
    vtss\_phy\_10g\_ckout\_conf\_t, 248  
    vtss\_phy\_10g\_sckout\_conf\_t, 312  
squelch\_on\_lopc  
    vtss\_phy\_10g\_rxckout\_conf\_t, 311  
squelch\_on\_pcs\_fault  
    vtss\_phy\_10g\_rxckout\_conf\_t, 310  
squelsh  
    vtss\_sync\_clock\_in\_t, 499  
srate  
    vtss\_phy\_10g\_pkt\_gen\_conf\_t, 300  
src  
    vtss\_phy\_10g\_ckout\_conf\_t, 248  
    vtss\_phy\_10g\_sckout\_conf\_t, 312  
    vtss\_phy\_clock\_conf\_t, 332  
src\_ip  
    vtss\_phy\_ts\_fifo\_sig\_t, 380  
src\_port\_identity  
    vtss\_phy\_ts\_fifo\_sig\_t, 379  
sref\_clk\_div  
    vtss\_phy\_10g\_mode\_t, 284  
stack\_depth  
    vtss\_phy\_ts\_mpls\_conf\_t, 395  
stack\_level  
    vtss\_phy\_ts\_mpls\_conf\_t, 397  
stack\_port\_a  
    vtss\_vstax\_route\_entry\_t, 549  
stack\_port\_b  
    vtss\_vstax\_route\_entry\_t, 549  
stack\_queue  
    vtss\_packet\_rx\_queue\_map\_t, 219  
stack\_ref\_point  
    vtss\_phy\_ts\_mpls\_conf\_t, 395  
static\_conf  
    vtss\_ewis\_conf\_s, 102  
status  
    vtss\_phy\_10g\_status\_t, 322  
    vtss\_phy\_veriphy\_result\_t, 418  
stripped\_tag  
    vtss\_packet\_rx\_info\_t, 205  
stuck\_at\_01  
    vtss\_phy\_10g\_prbs\_mon\_conf\_t, 309  
stuck\_at\_par  
    vtss\_phy\_10g\_prbs\_mon\_conf\_t, 308  
suspend\_tick\_cnt  
    vtss\_fdma\_throttle\_cfg\_t, 152  
sw\_tstamp  
    tag\_vtss\_fdma\_list, 38  
    vtss\_packet\_rx\_info\_t, 207  
    vtss\_packet\_rx\_meta\_t, 214  
switch\_frm  
    vtss\_packet\_tx\_info\_t, 224  
sym  
    vtss\_packet\_tx\_info\_t, 228  
symmetric\_pause

vtss\_phy\_10g\_clause\_37\_adv\_t, 250  
 vtss\_phy\_aneg\_t, 331  
 vtss\_port\_clause\_37\_adv\_t, 427

**TRUE**  
 types.h, 601

**table**  
 vtss\_vstax\_route\_table\_t, 550

**tag**  
 vtss\_ece\_key\_t, 82  
 vtss\_mirror\_conf\_t, 189  
 vtss\_packet\_rx\_header\_t, 201  
 vtss\_packet\_rx\_info\_t, 204  
 vtss\_packet\_tx\_info\_t, 225  
 vtss\_qce\_key\_t, 469  
 vtss\_vce\_key\_t, 530

**tag\_class\_enable**  
 vtss\_qos\_port\_conf\_t, 479

**tag\_default\_dei**  
 vtss\_qos\_port\_conf\_t, 481

**tag\_default\_pcp**  
 vtss\_qos\_port\_conf\_t, 481

**tag\_dei\_map**  
 vtss\_qos\_port\_conf\_t, 482

**tag\_pcp\_map**  
 vtss\_qos\_port\_conf\_t, 481

**tag\_range\_mode**  
 vtss\_phy\_ts\_eth\_conf\_t, 374

**tag\_remark\_mode**  
 vtss\_qos\_port\_conf\_t, 481

**tag\_type**  
 vtss\_packet\_rx\_info\_t, 204

**tag\_vtss\_fdma\_list**, 36

act\_len, 37  
 afi\_frm\_cnt, 39  
 afi\_seq\_number, 39  
 alloc\_ptr, 38  
 frm\_ptr, 37  
 next, 39  
 rx\_info, 38  
 sw\_tstamp, 38  
 user, 38

**tagged**  
 vtss\_ece\_tag\_t, 88  
 vtss\_qce\_tag\_t, 474  
 vtss\_vce\_tag\_t, 535

**tagprio**  
 vtss\_tci\_t, 502

**target**  
 vtss\_inst\_create\_t, 164

**tbi**  
 vtss\_phy\_reset\_conf\_t, 355

**tc\_op\_mode**  
 vtss\_phy\_ts\_init\_conf\_t, 389

**tci**  
 vtss\_vstax\_tx\_header\_t, 553

**tcp\_ack**  
 vtss\_ace\_frame\_ipv4\_t, 47  
 vtss\_ace\_frame\_ipv6\_t, 51

**tcp\_fin**  
 vtss\_ace\_frame\_ipv4\_t, 46  
 vtss\_ace\_frame\_ipv6\_t, 50

**tcp\_psh**  
 vtss\_ace\_frame\_ipv4\_t, 47  
 vtss\_ace\_frame\_ipv6\_t, 51

**tcp\_rst**  
 vtss\_ace\_frame\_ipv4\_t, 47  
 vtss\_ace\_frame\_ipv6\_t, 51

**tcp\_syn**  
 vtss\_ace\_frame\_ipv4\_t, 47  
 vtss\_ace\_frame\_ipv6\_t, 51

**tcp\_urg**  
 vtss\_ace\_frame\_ipv4\_t, 47  
 vtss\_ace\_frame\_ipv6\_t, 51

**test\_conf**  
 vtss\_ewis\_conf\_s, 103

**test\_pattern\_ana**  
 vtss\_ewis\_test\_conf\_s, 130

**test\_pattern\_gen**  
 vtss\_ewis\_test\_conf\_s, 130

**thrd\_lvl\_after\_top**  
 vtss\_phy\_ts\_mpls\_conf\_t, 396

**thrd\_lvl\_before\_end**  
 vtss\_phy\_ts\_mpls\_conf\_t, 397

**timeout**  
 vtss\_mtimer\_t, 190

**timestamp**  
 vtss\_phy\_10g\_timestamp\_val\_t, 324

**to\_cpu**  
 vtss\_policer\_ext\_t, 423

**tod\_get\_ns\_cnt\_cb\_t**  
 vtss\_misc\_api.h, 759

**top**  
 vtss\_phy\_ts\_mpls\_conf\_t, 396

**top\_down**  
 vtss\_phy\_ts\_mpls\_conf\_t, 396

**topology\_type**  
 vtss\_vstax\_route\_table\_t, 550

**tpid**  
 vtss\_packet\_port\_filter\_t, 196  
 vtss\_phy\_ts\_eth\_conf\_t, 372  
 vtss\_vlan\_tag\_t, 541

**train**  
 vtss\_phy\_10g\_base\_kr\_status\_t, 244

**training**  
 vtss\_phy\_10g\_base\_kr\_train\_aneg\_t, 245

**trans\_vid**  
 vtss\_vlan\_trans\_grp2vlan\_conf\_t, 542

**trig\_ch\_id**  
 vtss\_phy\_10g\_auto\_failover\_conf\_t, 237

**trmthd\_c0**  
 vtss\_phy\_10g\_base\_kr\_training\_t, 247

**trmthd\_cm**  
 vtss\_phy\_10g\_base\_kr\_training\_t, 247

**trmthd\_cp**  
 vtss\_phy\_10g\_base\_kr\_training\_t, 246

**ts**

vtss\_ts\_timestamp\_t, 510  
ts\_fifo\_drop\_cnt  
    vtss\_phy\_ts\_stats\_t, 413  
ts\_fifo\_tx\_cnt  
    vtss\_phy\_ts\_stats\_t, 413  
ts\_format  
    vtss\_phy\_ts\_ietf\_mpls\_ach\_oam\_conf\_t, 386  
ts\_id  
    vtss\_ts\_id\_t, 506  
ts\_offset  
    vtss\_phy\_ts\_generic\_action\_t, 384  
ts\_type  
    vtss\_phy\_ts\_generic\_action\_t, 384  
ts\_valid  
    vtss\_ts\_timestamp\_t, 510  
tstamp\_id  
    vtss\_packet\_rx\_info\_t, 207  
tstamp\_id\_decoded  
    vtss\_packet\_rx\_info\_t, 207  
tstpat\_cnt  
    vtss\_ewis\_test\_status\_s, 131  
tti  
    vtss\_ewis\_tti\_s, 132  
ttl  
    vtss\_ace\_frame\_ipv4\_t, 44  
    vtss\_ace\_frame\_ipv6\_t, 49  
    vtss\_vstax\_port\_conf\_t, 548  
    vtss\_vstax\_tx\_header\_t, 553  
tx\_b1  
    vtss\_ewis\_tx\_passthru\_s, 137  
tx\_b2  
    vtss\_ewis\_tx\_passthru\_s, 138  
tx\_buf\_cnt  
    vtss\_fdma\_cfg\_t, 145  
tx\_c2  
    vtss\_ewis\_tx\_oh\_s, 135  
tx\_ch  
    vtss\_phy\_10g\_lane\_sync\_conf\_t, 278  
tx\_clk\_skew\_ps  
    vtss\_phy\_rgmii\_conf\_t, 357  
tx\_dcc\_l  
    vtss\_ewis\_tx\_oh\_s, 134  
    vtss\_ewis\_tx\_passthru\_s, 139  
tx\_dcc\_s  
    vtss\_ewis\_tx\_oh\_s, 133  
    vtss\_ewis\_tx\_passthru\_s, 137  
tx\_discard  
    vtss\_evc\_counters\_t, 97  
tx\_done\_cb  
    vtss\_fdma\_cfg\_t, 146  
tx\_e1  
    vtss\_ewis\_tx\_oh\_s, 133  
    vtss\_ewis\_tx\_passthru\_s, 137  
tx\_e2  
    vtss\_ewis\_tx\_oh\_s, 134  
    vtss\_ewis\_tx\_passthru\_s, 139  
tx\_etherStatsBroadcastPkts  
    vtss\_port\_rmon\_counters\_t, 453  
tx\_etherStatsCollisions  
    vtss\_port\_rmon\_counters\_t, 453  
tx\_etherStatsDropEvents  
    vtss\_port\_rmon\_counters\_t, 452  
tx\_etherStatsMulticastPkts  
    vtss\_port\_rmon\_counters\_t, 453  
tx\_etherStatsOctets  
    vtss\_port\_rmon\_counters\_t, 452  
tx\_etherStatsPkts  
    vtss\_port\_rmon\_counters\_t, 452  
tx\_etherStatsPkts1024to1518Octets  
    vtss\_port\_rmon\_counters\_t, 454  
tx\_etherStatsPkts128to255Octets  
    vtss\_port\_rmon\_counters\_t, 454  
tx\_etherStatsPkts1519toMaxOctets  
    vtss\_port\_rmon\_counters\_t, 454  
tx\_etherStatsPkts256to511Octets  
    vtss\_port\_rmon\_counters\_t, 454  
tx\_etherStatsPkts512to1023Octets  
    vtss\_port\_rmon\_counters\_t, 454  
tx\_etherStatsPkts64Octets  
    vtss\_port\_rmon\_counters\_t, 453  
tx\_etherStatsPkts65to127Octets  
    vtss\_port\_rmon\_counters\_t, 453  
tx\_f1  
    vtss\_ewis\_tx\_oh\_s, 133  
    vtss\_ewis\_tx\_passthru\_s, 137  
tx\_f2  
    vtss\_ewis\_tx\_oh\_s, 135  
tx\_fault  
    vtss\_sublayer\_status\_t, 498  
tx\_fifo\_hi\_clk\_cycs  
    vtss\_phy\_ts\_init\_conf\_t, 388  
tx\_fifo\_lo\_clk\_cycs  
    vtss\_phy\_ts\_init\_conf\_t, 389  
tx\_fifo\_mode  
    vtss\_phy\_ts\_init\_conf\_t, 388  
tx\_fifo\_spi\_conf  
    vtss\_phy\_ts\_init\_conf\_t, 388  
tx\_frames  
    vtss\_basic\_counters\_t, 68  
tx\_green  
    vtss\_evc\_counters\_t, 97  
tx\_j0  
    vtss\_ewis\_tx\_passthru\_s, 136  
tx\_k1  
    vtss\_ewis\_tx\_passthru\_s, 138  
tx\_k1\_k2  
    vtss\_ewis\_tx\_oh\_s, 134  
tx\_k2  
    vtss\_ewis\_tx\_passthru\_s, 138  
tx\_loh  
    vtss\_ewis\_tx\_passthru\_s, 139  
tx\_macro  
    vtss\_phy\_10g\_lane\_sync\_conf\_t, 277  
tx\_n1  
    vtss\_ewis\_tx\_oh\_s, 135  
tx\_oh

vtss\_ewis\_conf\_s, 103  
 tx\_oh\_passthru  
     vtss\_ewis\_conf\_s, 103  
 tx\_out\_bytes  
     vtss\_eee\_port\_counter\_t, 92  
 tx\_out\_bytes\_get  
     vtss\_eee\_port\_counter\_t, 91  
 tx\_prio  
     vtss\_port\_proprietary\_counters\_t, 447  
 tx\_reil  
     vtss\_ewis\_tx\_passthru\_s, 138  
 tx\_remote\_fault  
     vtss\_phy\_aneg\_t, 332  
 tx\_s1  
     vtss\_ewis\_tx\_oh\_s, 134  
     vtss\_ewis\_tx\_passthru\_s, 139  
 tx\_soh  
     vtss\_ewis\_tx\_passthru\_s, 138  
 tx\_ts\_len  
     vtss\_phy\_ts\_init\_conf\_t, 388  
 tx\_tw  
     vtss\_eee\_port\_conf\_t, 90  
 tx\_vstax\_hdr  
     vtss\_packet\_tx\_info\_t, 227  
 tx\_yellow  
     vtss\_evc\_counters\_t, 97  
 tx\_z0  
     vtss\_ewis\_tx\_oh\_s, 134  
     vtss\_ewis\_tx\_passthru\_s, 137  
 tx\_z1\_z2  
     vtss\_ewis\_tx\_oh\_s, 135  
     vtss\_ewis\_tx\_passthru\_s, 139  
 tx\_z3\_z4  
     vtss\_ewis\_tx\_oh\_s, 135  
 type  
     vtss\_ace\_t, 56  
     vtss\_dlb\_policer\_conf\_t, 75  
     vtss\_ece\_inner\_tag\_t, 80  
     vtss\_ece\_key\_t, 83  
     vtss\_eps\_port\_conf\_t, 93  
     vtss\_fan\_conf\_t, 141  
     vtss\_ip\_addr\_t, 165  
     vtss\_phy\_10g\_id\_t, 269  
     vtss\_qce\_key\_t, 469  
     vtss\_routing\_entry\_t, 489  
     vtss\_sflow\_port\_conf\_t, 493  
     vtss\_vcap\_vr\_t, 522  
     vtss\_vce\_key\_t, 531  
 types.h  
     BOOL, 620  
     FALSE, 602  
     i16, 619  
     i32, 619  
     i64, 619  
     i8, 619  
     MAC\_ADDR\_BROADCAST, 610  
     PRIx64, 600  
     PRIu64, 600  
     PRId64, 600  
     TRUE, 601  
     u16, 620  
     u32, 620  
     u64, 620  
     u8, 619  
     uintptr\_t, 620  
     VTSS\_ACL\_POLICER\_NO\_END, 614  
     VTSS\_ACL\_POLICER\_NO\_START, 614  
     VTSS\_ACL\_POLICERS, 614  
     VTSS\_ACL\_POLICIES, 615  
     VTSS\_ACL\_POLICY\_NO\_END, 616  
     VTSS\_ACL\_POLICY\_NO\_MAX, 615  
     VTSS\_ACL\_POLICY\_NO\_MIN, 615  
     VTSS\_ACL\_POLICY\_NO\_NONE, 615  
     VTSS\_ACL\_POLICY\_NO\_START, 615  
     VTSS\_AGGR\_NO\_END, 611  
     VTSS\_AGGR\_NO\_NONE, 611  
     VTSS\_AGGR\_NO\_START, 611  
     VTSS\_AGGRS, 610  
     VTSS\_BIT64, 600  
     VTSS\_BITMASK64, 600  
     VTSS\_BITRATE\_DISABLED, 608  
     VTSS\_CLOCK\_IDENTITY\_LENGTH, 618  
     VTSS\_DEI\_ARRAY\_SIZE, 607  
     VTSS\_DEI\_END, 607  
     VTSS\_DEI\_START, 607  
     VTSS\_DEIS, 606  
     VTSS\_DPL\_ARRAY\_SIZE, 608  
     VTSS\_DPL\_END, 608  
     VTSS\_DPL\_START, 608  
     VTSS\_DPLS, 607  
     VTSS\_ENCODE\_BITFIELD64, 601  
     VTSS\_ENCODE\_BITMASK64, 601  
     VTSSETYPE\_VTSS, 609  
     VTSS\_EVCS, 610  
     VTSS\_EXTRACT\_BITFIELD64, 601  
     VTSS\_GLAG\_NO\_END, 612  
     VTSS\_GLAG\_NO\_NONE, 611  
     VTSS\_GLAG\_NO\_START, 612  
     VTSS\_GLAG\_PORT\_ARRAY\_SIZE, 613  
     VTSS\_GLAG\_PORT\_END, 612  
     VTSS\_GLAG\_PORT\_START, 612  
     VTSS\_GLAG\_PORTS, 612  
     VTSS\_GLAGS, 611  
     VTSS\_HQOS\_COUNT, 616  
     VTSS\_HQOS\_ID\_NONE, 616  
     VTSS\_INTERVAL\_MS, 617  
     VTSS\_INTERVAL\_NS, 617  
     VTSS\_INTERVAL\_PS, 618  
     VTSS\_INTERVAL\_SEC, 617  
     VTSS\_INTERVAL\_US, 617  
     VTSS\_ISDX\_NONE, 610  
     VTSS\_MAC\_ADDR\_SZ\_BYTES, 610  
     VTSS\_MAX\_TIMEINTERVAL, 617  
     VTSS\_ONE\_MILL, 616  
     VTSS\_ONE\_MIA, 616  
     VTSS\_PACKET\_RATE\_DISABLED, 602

VTSS\_PACKET\_RX\_GRP\_CNT, 613  
VTSS\_PACKET\_RX\_QUEUE\_CNT, 613  
VTSS\_PACKET\_RX\_QUEUE\_END, 614  
VTSS\_PACKET\_RX\_QUEUE\_NONE, 613  
VTSS\_PACKET\_RX\_QUEUE\_START, 614  
VTSS\_PACKET\_TX\_GRP\_CNT, 613  
VTSS\_PCP\_ARRAY\_SIZE, 606  
VTSS\_PCP\_END, 606  
VTSS\_PCP\_START, 606  
VTSS\_PCPS, 606  
VTSS\_PORT\_ARRAY\_SIZE, 603  
VTSS\_PORT\_COUNT, 602  
VTSS\_PORT\_IS\_PORT, 604  
VTSS\_PORT\_NO\_CPU, 603  
VTSS\_PORT\_NO\_END, 603  
VTSS\_PORT\_NO\_NONE, 603  
VTSS\_PORT\_NO\_START, 603  
VTSS\_PORTS, 602  
VTSS\_PRIO\_ARRAY\_SIZE, 605  
VTSS\_PRIO\_END, 604  
VTSS\_PRIO\_NO\_NONE, 604  
VTSS\_PRIO\_START, 604  
VTSS\_PRIOS, 604  
VTSS\_QUEUE\_ARRAY\_SIZE, 605  
VTSS\_QUEUE\_END, 605  
VTSS\_QUEUE\_START, 605  
VTSS\_QUEUES, 605  
VTSS\_SEC\_NS\_INTERVAL, 618  
VTSS\_SYNCE\_CLK\_PORT\_ARRAY\_SIZE, 618  
VTSS\_VID\_ALL, 609  
VTSS\_VID\_DEFAULT, 609  
VTSS\_VID\_NULL, 608  
VTSS\_VID\_RESERVED, 609  
VTSS\_VIDS, 609  
vtss\_ece\_dir\_t, 628  
vtss\_ece\_inner\_tag\_type\_t, 629  
vtss\_ece\_pop\_tag\_t, 629  
vtss\_hqos\_sch\_mode\_t, 630  
vtss\_ip\_type\_t, 627  
vtss\_isdx\_t, 621  
vtss\_mac\_addr\_t, 621  
vtss\_mem\_flags\_t, 624  
vtss\_packet\_reg\_type\_t, 626  
vtss\_packet\_rx\_grp\_t, 621  
vtss\_packet\_tx\_grp\_t, 621  
vtss\_policer\_type\_t, 626  
vtss\_port\_interface\_t, 624  
vtss\_serdes\_mode\_t, 625  
vtss\_storm\_policer\_mode\_t, 626  
vtss\_vcap\_bit\_t, 627  
vtss\_vcap\_key\_type\_t, 628  
vtss\_vcap\_vr\_type\_t, 628  
vtss\_vdd\_t, 627  
vtss\_vlan\_frame\_t, 626

u16  
types.h, 620

u32  
types.h, 620

u64  
types.h, 620

u8  
types.h, 619

UNSIGNED\_STORAGE\_COUNT  
vtss\_phy\_10g\_api.h, 847

uc\_no\_flood  
vtss\_policer\_ext\_t, 423

uint  
vtss\_os\_custom.h, 787

uintptr\_t  
types.h, 620

ulong  
vtss\_os\_custom.h, 787

uncorrected\_block\_cnt  
vtss\_phy\_10g\_kr\_status\_fec\_t, 275

unicast  
vtss\_policer\_ext\_t, 422

unidir  
vtss\_phy\_conf\_t, 336

unidirectional\_ability  
vtss\_port\_status\_t, 459

unknown  
vtss\_ace\_frame\_arp\_t, 41

untagged\_vid  
vtss\_vlan\_port\_conf\_t, 539

update  
vtss\_phy\_10g\_pkt\_mon\_conf\_t, 301

upper  
vtss\_phy\_ts\_eth\_conf\_t, 374  
vtss\_phy\_ts\_mpls\_lvl\_rng\_t, 398  
vtss\_phy\_ts\_ptp\_conf\_t, 406  
vtss\_phy\_ts\_y1731\_oam\_conf\_t, 415

upsid  
vtss\_mac\_table\_entry\_t, 186  
vtss\_vstax\_glag\_entry\_t, 547  
vtss\_vstax\_rx\_header\_t, 551  
vtss\_vstax\_tx\_header\_t, 553

upsid\_0  
vtss\_vstax\_conf\_t, 545

upsid\_1  
vtss\_vstax\_conf\_t, 546

upspn  
vtss\_mac\_table\_entry\_t, 186  
vtss\_vstax\_glag\_entry\_t, 547

usage  
vtss\_fdma\_ch\_cfg\_t, 148

use\_as\_intrpt  
vtss\_gpio\_10g\_gpio\_mode\_t, 158

use\_conf  
vtss\_phy\_10g\_mode\_t, 285

user  
tag\_vtss\_fdma\_list, 38

usr\_prio  
vtss\_ace\_vlan\_t, 59  
vtss\_qos\_port\_conf\_t, 479

v  
vtss\_vcap\_vr\_t, 522

v3  
     vtss\_phy\_10g\_ob\_status\_t, 294

v4  
     vtss\_phy\_10g\_ob\_status\_t, 294

v5  
     vtss\_phy\_10g\_ob\_status\_t, 295

v\_gpio  
     vtss\_phy\_10g\_auto\_failover\_conf\_t, 237

VIPER\_B\_FIFO\_RESET  
     options.h, 574

VTSS\_10G\_PHY\_GPIO\_MAX  
     vtss\_phy\_10g\_api.h, 849

VTSS\_10G\_PHY\_GPIO\_MAX  
     vtss\_phy\_10g\_api.h, 849

VTSS\_ACE\_ID\_LAST  
     vtss\_security\_api.h, 1102

VTSS\_ACL\_POLICER\_NO\_END  
     types.h, 614

VTSS\_ACL\_POLICER\_NO\_START  
     types.h, 614

VTSS\_ACL\_POLICERS  
     types.h, 614

VTSS\_ACL\_POLICIES  
     types.h, 615

VTSS\_ACL\_POLICY\_NO\_END  
     types.h, 616

VTSS\_ACL\_POLICY\_NO\_MAX  
     types.h, 615

VTSS\_ACL\_POLICY\_NO\_MIN  
     types.h, 615

VTSS\_ACL\_POLICY\_NO\_NONE  
     types.h, 615

VTSS\_ACL\_POLICY\_NO\_START  
     types.h, 615

VTSS\_AFI\_FPS\_MAX  
     vtss\_fdma\_api.h, 646

VTSS\_AGGR\_NO\_END  
     types.h, 611

VTSS\_AGGR\_NO\_NONE  
     types.h, 611

VTSS\_AGGR\_NO\_START  
     types.h, 611

VTSS\_AGGRS  
     types.h, 610

VTSS\_ARCH\_JAGUAR\_1  
     options.h, 560

VTSS\_ARCH\_JAGUAR\_1\_CE\_SWITCH  
     options.h, 560

VTSS\_ARCH\_MALIBU\_B  
     options.h, 577

VTSS\_ARCH\_MALIBU  
     options.h, 577

VTSS\_ARCH\_VENICE\_C  
     options.h, 577

VTSS\_ARP\_CNT  
     vtss\_l3\_api.h, 744

VTSS\_ARP\_IPV4\_RELATIONS  
     vtss\_l3\_api.h, 745

VTSS\_ARP\_IPV6\_RELATIONS  
     vtss\_l3\_api.h, 745

VTSS\_BIT64  
     types.h, 600

VTSS\_BITMASK64  
     types.h, 600

VTSS\_BITRATE\_DISABLED  
     types.h, 608

VTSS\_CHIP\_10G\_PHY  
     options.h, 562

VTSS\_CHIP CU PHY  
     options.h, 574

VTSS\_CLOCK\_IDENTITY\_LENGTH  
     types.h, 618

VTSS\_DEI\_ARRAY\_SIZE  
     types.h, 607

VTSS\_DEI\_END  
     types.h, 607

VTSS\_DEI\_START  
     types.h, 607

VTSS\_DEIS  
     types.h, 606

VTSS\_DIV64  
     vtss\_os\_custom.h, 788

VTSS\_DPL\_ARRAY\_SIZE  
     types.h, 608

VTSS\_DPL\_END  
     types.h, 608

VTSS\_DPL\_START  
     types.h, 608

VTSS\_DPLS  
     types.h, 607

VTSS\_ECE\_ID\_LAST  
     vtss\_evc\_api.h, 634

VTSS\_ENCODE\_BITFIELD64  
     types.h, 601

VTSS\_ENCODE\_BITMASK64  
     types.h, 601

VTSS\_ERPI\_ARRAY\_SIZE  
     vtss\_l2\_api.h, 692

VTSS\_ERPI\_END  
     vtss\_l2\_api.h, 692

VTSS\_ERPI\_START  
     vtss\_l2\_api.h, 691

VTSS\_ERPIS  
     vtss\_l2\_api.h, 691

VTSSETYPE\_VTSS  
     types.h, 609

VTSS\_EVC\_ID\_NONE  
     vtss\_evc\_api.h, 634

VTSS\_EVC\_POLICER\_ID\_DISCARD  
     vtss\_evc\_api.h, 634

VTSS\_EVC\_POLICER\_ID\_EVC  
     vtss\_evc\_api.h, 634

VTSS\_EVC\_POLICER\_ID\_NONE  
     vtss\_evc\_api.h, 634

VTSS\_EVC\_POLICERS  
vtss\_evc\_api.h, 633

VTSS\_EVCS  
types.h, 610

VTSS\_EWIS\_AISL\_EV  
vtss\_wis\_api.h, 1138

VTSS\_EWIS\_AISP\_EV  
vtss\_wis\_api.h, 1138

VTSS\_EWIS\_B1\_NZ\_EV  
vtss\_wis\_api.h, 1141

VTSS\_EWIS\_B1\_THRESH\_EV  
vtss\_wis\_api.h, 1142

VTSS\_EWIS\_B2\_NZ\_EV  
vtss\_wis\_api.h, 1141

VTSS\_EWIS\_B2\_THRESH\_EV  
vtss\_wis\_api.h, 1142

VTSS\_EWIS\_B3\_NZ\_EV  
vtss\_wis\_api.h, 1141

VTSS\_EWIS\_B3\_THRESH\_EV  
vtss\_wis\_api.h, 1142

VTSS\_EWIS\_FAIS\_EV  
vtss\_wis\_api.h, 1137

VTSS\_EWIS\_FERDIP\_EV  
vtss\_wis\_api.h, 1140

VTSS\_EWIS\_FEUNEQP\_EV  
vtss\_wis\_api.h, 1140

VTSS\_EWIS\_FPLM\_EV  
vtss\_wis\_api.h, 1137

VTSS\_EWIS\_HIGH\_BER\_EV  
vtss\_wis\_api.h, 1140

VTSS\_EWIS\_LCDP\_EV  
vtss\_wis\_api.h, 1138

VTSS\_EWIS\_LOF\_EV  
vtss\_wis\_api.h, 1137

VTSS\_EWIS\_LOPC\_EV  
vtss\_wis\_api.h, 1139

VTSS\_EWIS\_LOPP\_EV  
vtss\_wis\_api.h, 1138

VTSS\_EWIS\_LOS\_EV  
vtss\_wis\_api.h, 1137

VTSS\_EWIS\_MODULE\_EV  
vtss\_wis\_api.h, 1139

VTSS\_EWIS\_PCS\_RECEIVE\_FAULT\_PEND  
vtss\_wis\_api.h, 1141

VTSS\_EWIS\_PLMP\_EV  
vtss\_wis\_api.h, 1138

VTSS\_EWIS\_RDIL\_EV  
vtss\_wis\_api.h, 1137

VTSS\_EWIS\_REIL\_EV  
vtss\_wis\_api.h, 1140

VTSS\_EWIS\_REIL\_NZ\_EV  
vtss\_wis\_api.h, 1141

VTSS\_EWIS\_REIL\_THRESH\_EV  
vtss\_wis\_api.h, 1142

VTSS\_EWIS\_REIP\_EV  
vtss\_wis\_api.h, 1140

VTSS\_EWIS\_REIP\_NZ\_EV  
vtss\_wis\_api.h, 1142

VTSS\_EWIS\_RXLOL\_EV  
vtss\_wis\_api.h, 1139

VTSS\_EWIS\_SEF\_EV  
vtss\_wis\_api.h, 1136

VTSS\_EWIS\_TXLOL\_EV  
vtss\_wis\_api.h, 1139

VTSS\_EWIS\_UNEQP\_EV  
vtss\_wis\_api.h, 1139

VTSS\_EXTRACT\_BITFIELD64  
types.h, 601

VTSS\_FDMA\_CCM\_FPS\_MAX  
vtss\_fdma\_api.h, 646

VTSS\_FDMA\_CCM\_FREQ\_LIST\_LEN  
vtss\_fdma\_api.h, 646

VTSS\_FDMA\_CCM\_QUOTIENT\_MAX  
vtss\_fdma\_api.h, 646

VTSS\_FDMA\_CH\_CNT  
vtss\_fdma\_api.h, 643

VTSS\_FDMA\_DCB\_SIZE\_BYTES  
vtss\_fdma\_api.h, 644

VTSS\_FDMA\_HDR\_SIZE\_BYTES  
vtss\_fdma\_api.h, 644

VTSS\_FDMA\_MAX\_DATA\_PER\_DCB\_BYTES  
vtss\_fdma\_api.h, 644

VTSS\_FDMA\_MAX\_FRAME\_SIZE\_BYTES  
vtss\_fdma\_api.h, 645

VTSS\_FDMA\_MIN\_DATA\_PER\_INJ\_SOF\_DCB\_BYTES  
vtss\_fdma\_api.h, 645

VTSS\_FDMA\_MIN\_DATA\_PER\_NON\_SOF\_DCB\_BYTES  
vtss\_fdma\_api.h, 645

VTSS\_FDMA\_MIN\_DATA\_PER\_XTR\_SOF\_DCB\_BYTES  
vtss\_fdma\_api.h, 645

VTSS\_FDMA\_MIN\_FRAME\_SIZE\_BYTES  
vtss\_fdma\_api.h, 644

VTSS\_FEATURE\_10GBASE\_KR  
options.h, 572

VTSS\_FEATURE\_10G  
options.h, 563

VTSS\_FEATURE\_ACL\_V1  
options.h, 571

VTSS\_FEATURE\_ACL  
options.h, 571

VTSS\_FEATURE\_AFI\_FDMA  
options.h, 573

VTSS\_FEATURE\_AGGR\_GLAG  
options.h, 561

VTSS\_FEATURE\_CLAUSE\_37  
options.h, 563

VTSS\_FEATURE\_E\_TREE  
options.h, 560

VTSS\_FEATURE\_EDC\_FW\_LOAD  
options.h, 576

VTSS\_FEATURE\_EEE

VTSS\_FEATURE\_EVC  
 options.h, 560  
 VTSS\_FEATURE\_EXC\_COL\_CONT  
 options.h, 564  
 VTSS\_FEATURE\_FAN  
 options.h, 562, 572  
 VTSS\_FEATURE\_FDMA  
 options.h, 573  
 VTSS\_FEATURE\_IRQ\_CONTROL  
 options.h, 573  
 VTSS\_FEATURE\_LAYER2  
 options.h, 569  
 VTSS\_FEATURE\_LAYER3  
 options.h, 570  
 VTSS\_FEATURE\_LED\_POW\_REDUC  
 options.h, 572  
 VTSS\_FEATURE\_MAC\_AGE\_AUTO  
 options.h, 570  
 VTSS\_FEATURE\_MAC\_CPU\_QUEUE  
 options.h, 570  
 VTSS\_FEATURE\_MISC  
 options.h, 562  
 VTSS\_FEATURE\_NPI  
 options.h, 564  
 VTSS\_FEATURE\_PACKET\_GROUPING  
 options.h, 569  
 VTSS\_FEATURE\_PACKET\_PORT\_REG  
 options.h, 569  
 VTSS\_FEATURE\_PACKET\_RX  
 options.h, 568  
 VTSS\_FEATURE\_PACKET\_TX  
 options.h, 568  
 VTSS\_FEATURE\_PACKET  
 options.h, 568  
 VTSS\_FEATURE\_PHY\_TIMESTAMP  
 options.h, 561  
 VTSS\_FEATURE\_PORT\_CNT\_BRIDGE  
 options.h, 564  
 VTSS\_FEATURE\_PORT\_CNT\_ETHER\_LIKE  
 options.h, 564  
 VTSS\_FEATURE\_PORT\_CONTROL  
 options.h, 563  
 VTSS\_FEATURE\_PORT\_IFH  
 options.h, 563  
 VTSS\_FEATURE\_PORT\_MUX  
 options.h, 570  
 VTSS\_FEATURE\_PVLAN  
 options.h, 562  
 VTSS\_FEATURE\_QCL\_V2  
 options.h, 565  
 VTSS\_FEATURE\_QCL  
 options.h, 565  
 VTSS\_FEATURE\_QOS\_CLASSIFICATION\_V2  
 options.h, 567  
 VTSS\_FEATURE\_QOS\_DSCP\_REMARK\_V2  
 options.h, 568  
 VTSS\_FEATURE\_QOS\_DSCP\_REMARK  
 options.h, 567  
 VTSS\_FEATURE\_QOS\_EGRESS\_QUEUE\_SHAPE\_RS\_EB  
 options.h, 567  
 VTSS\_FEATURE\_QOS\_EGRESS\_QUEUE\_SHAPE\_RS  
 options.h, 567  
 VTSS\_FEATURE\_QOS\_POLICER\_DLBB  
 options.h, 574  
 VTSS\_FEATURE\_QOS\_PORT\_POLICER\_EXT\_DPBL  
 options.h, 566  
 VTSS\_FEATURE\_QOS\_PORT\_POLICER\_EXT\_FPS  
 options.h, 565  
 VTSS\_FEATURE\_QOS\_PORT\_POLICER\_EXT\_FC  
 options.h, 565  
 VTSS\_FEATURE\_QOS\_PORT\_POLICER\_EXT\_TTM  
 options.h, 566  
 VTSS\_FEATURE\_QOS\_PORT\_POLICER\_EXT  
 options.h, 565  
 VTSS\_FEATURE\_QOS\_QUEUE\_POLICER  
 options.h, 566  
 VTSS\_FEATURE\_QOS\_QUEUE\_TX  
 options.h, 566  
 VTSS\_FEATURE\_QOS\_SCHEDULER\_V2  
 options.h, 566  
 VTSS\_FEATURE\_QOS\_TAG\_REMARK\_V2  
 options.h, 567  
 VTSS\_FEATURE\_QOS\_WRED  
 options.h, 568  
 VTSS\_FEATURE\_QOS  
 options.h, 564  
 VTSS\_FEATURE\_SERDES\_MACRO\_SETTINGS  
 options.h, 572  
 VTSS\_FEATURE\_SERIAL\_GPIO  
 options.h, 563  
 VTSS\_FEATURE\_SFLOW  
 options.h, 574  
 VTSS\_FEATURE\_SYNCE\_10G  
 options.h, 576  
 VTSS\_FEATURE\_SYNCE  
 options.h, 571  
 VTSS\_FEATURE\_TIMESTAMP  
 options.h, 560  
 VTSS\_FEATURE\_VCAP  
 options.h, 571, 577  
 VTSS\_FEATURE\_VCL  
 options.h, 571  
 VTSS\_FEATURE\_VLAN\_PORT\_V2  
 options.h, 569  
 VTSS\_FEATURE\_VLAN\_SVL  
 options.h, 570  
 VTSS\_FEATURE\_VLAN\_TRANSLATION  
 options.h, 573  
 VTSS\_FEATURE\_VLAN\_TX\_TAG  
 options.h, 569  
 VTSS\_FEATURE\_VSTAX\_V2  
 options.h, 561  
 VTSS\_FEATURE\_VSTAX

options.h, 561  
VTSS FEATURE WARM\_START  
options.h, 576  
VTSS FEATURE WIS  
options.h, 576  
VTSS FRAME GAP\_DEFAULT  
vtss\_port\_api.h, 1077  
VTSS\_GLAG\_NO\_END  
types.h, 612  
VTSS\_GLAG\_NO\_NONE  
types.h, 611  
VTSS\_GLAG\_NO\_START  
types.h, 612  
VTSS\_GLAG\_PORT\_ARRAY\_SIZE  
types.h, 613  
VTSS\_GLAG\_PORT\_END  
types.h, 612  
VTSS\_GLAG\_PORT\_START  
types.h, 612  
VTSS\_GLAG\_PORTS  
types.h, 612  
VTSS\_GLAGS  
types.h, 611  
VTSS\_HQOS\_COUNT  
types.h, 616  
VTSS\_HQOS\_ID\_NONE  
types.h, 616  
VTSS\_I2C\_NO\_MULTIPLEXER  
vtss\_init\_api.h, 662  
VTSS\_INTERVAL\_MS  
types.h, 617  
VTSS\_INTERVAL\_NS  
types.h, 617  
VTSS\_INTERVAL\_PS  
types.h, 618  
VTSS\_INTERVAL\_SEC  
types.h, 617  
VTSS\_INTERVAL\_US  
types.h, 617  
VTSS\_ISDX\_NONE  
types.h, 610  
VTSS\_JR1\_ARP\_CNT  
vtss\_l3\_api.h, 744  
VTSS\_JR1\_LPM\_CNT  
vtss\_l3\_api.h, 743  
VTSS\_JR1\_PACKET\_HDR\_SIZE\_BYTES  
vtss\_packet\_api.h, 812  
VTSS\_JR1\_RLEG\_CNT  
vtss\_l3\_api.h, 744  
VTSS\_JR1\_RX\_IFH\_SIZE  
vtss\_packet\_api.h, 813  
VTSS\_JR2\_PACKET\_HDR\_SIZE\_BYTES  
vtss\_packet\_api.h, 812  
VTSS\_JR2\_RX\_IFH\_SIZE  
vtss\_packet\_api.h, 814  
VTSS\_L26\_PACKET\_HDR\_SIZE\_BYTES  
vtss\_packet\_api.h, 813  
VTSS\_L26\_RX\_IFH\_SIZE  
options.h, 561  
VTSS\_PACKET\_API.h, 814  
VTSS\_LABS  
vtss\_os\_custom.h, 789  
vtss\_os\_ecos.h, 794  
vtss\_os\_linux.h, 805  
VTSS\_LLABS  
vtss\_os\_custom.h, 789  
vtss\_os\_ecos.h, 794  
vtss\_os\_linux.h, 806  
VTSS\_LPM\_CNT  
vtss\_l3\_api.h, 744  
VTSS\_MAC\_ADDR\_SZ\_BYTES  
types.h, 610  
VTSS\_MAC\_ADDRS  
vtss\_l2\_api.h, 684  
VTSS\_MAX\_FRAME\_LENGTH\_MAX  
vtss\_port\_api.h, 1078  
VTSS\_MAX\_FRAME\_LENGTH\_STANDARD  
vtss\_port\_api.h, 1078  
VTSS\_MAX\_TIMEINTERVAL  
types.h, 617  
VTSS\_MOD64  
vtss\_os\_custom.h, 788  
vtss\_os\_ecos.h, 794  
vtss\_os\_linux.h, 805  
VTSS\_MSLEEP  
vtss\_os\_custom.h, 787  
vtss\_os\_ecos.h, 792  
vtss\_os\_linux.h, 803  
VTSS\_MSTI\_ARRAY\_SIZE  
vtss\_l2\_api.h, 687  
VTSS\_MSTI\_END  
vtss\_l2\_api.h, 686  
VTSS\_MSTI\_START  
vtss\_l2\_api.h, 686  
VTSS\_MSTIS  
vtss\_l2\_api.h, 686  
VTSS\_MTIMER\_CANCEL  
vtss\_os\_custom.h, 788  
vtss\_os\_ecos.h, 793  
vtss\_os\_linux.h, 804  
VTSS\_MTIMER\_START  
vtss\_os\_custom.h, 788  
vtss\_os\_ecos.h, 793  
vtss\_os\_linux.h, 803  
VTSS\_MTIMER\_TIMEOUT  
vtss\_os\_custom.h, 788  
vtss\_os\_ecos.h, 793  
vtss\_os\_linux.h, 803  
VTSS\_NSLEEP  
vtss\_os\_ecos.h, 792  
vtss\_os\_linux.h, 802  
VTSS\_ONE\_MILL  
types.h, 616  
VTSS\_ONE\_MIA  
types.h, 616  
VTSS\_OPT\_FDMA\_DEBUG  
options.h, 575

VTSS\_OPT\_FDMA\_IRQ\_CONTEXT  
     options.h, 575

VTSS\_OPT\_TRACE  
     options.h, 575

VTSS\_OPT\_TS\_SPI\_FPGA  
     options.h, 562

VTSS\_OPT\_VAUI\_EQ\_CTRL  
     options.h, 575

VTSS\_OPT\_VCORE\_III  
     options.h, 573

VTSS\_OS\_BIG\_ENDIAN  
     vtss\_os\_ecos.h, 797  
     vtss\_os\_linux.h, 802

VTSS\_OS\_COMPILER\_ATTRIBUTE\_ALIGNED  
     vtss\_fdma\_api.h, 646  
     vtss\_os\_ecos.h, 796

VTSS\_OS\_CTZ64  
     vtss\_os\_custom.h, 789  
     vtss\_os\_ecos.h, 795  
     vtss\_os\_linux.h, 806

VTSS\_OS\_CTZ  
     vtss\_os\_custom.h, 789  
     vtss\_os\_ecos.h, 794  
     vtss\_os\_linux.h, 806

VTSS\_OS\_DCACHE\_FLUSH  
     vtss\_os\_ecos.h, 797

VTSS\_OS\_DCACHE\_INVALIDATE  
     vtss\_os\_ecos.h, 797

VTSS\_OS\_DCACHE\_LINE\_SIZE\_BYTES  
     vtss\_fdma\_api.h, 645  
     vtss\_os\_ecos.h, 796

VTSS\_OS\_FREE  
     vtss\_os\_custom.h, 790  
     vtss\_os\_ecos.h, 795  
     vtss\_os\_linux.h, 807

VTSS\_OS\_INTERRUPT\_DISABLE  
     vtss\_os\_ecos.h, 799

VTSS\_OS\_INTERRUPT\_FLAGS  
     vtss\_os\_ecos.h, 799

VTSS\_OS\_INTERRUPT\_RESTORE  
     vtss\_os\_ecos.h, 799

VTSS\_OS\_MALLOC  
     vtss\_os\_custom.h, 790  
     vtss\_os\_ecos.h, 795  
     vtss\_os\_linux.h, 807

VTSS\_OS\_NTOHL  
     vtss\_os\_ecos.h, 798  
     vtss\_os\_linux.h, 802

VTSS\_OS\_RAND  
     vtss\_os\_custom.h, 790  
     vtss\_os\_ecos.h, 796  
     vtss\_os\_linux.h, 807

VTSS\_OS\_REORDER\_BARRIER  
     vtss\_os\_ecos.h, 796

VTSS\_OS\_SCHEDULER\_FLAGS  
     vtss\_os\_ecos.h, 798  
     vtss\_os\_linux.h, 804

VTSS\_OS\_SCHEDULER\_LOCK  
     vtss\_os\_ecos.h, 798  
     vtss\_os\_linux.h, 804

VTSS\_OS\_SCHEDULER\_UNLOCK  
     vtss\_os\_ecos.h, 798  
     vtss\_os\_linux.h, 805

VTSS\_OS\_TIMESTAMP\_TYPE  
     vtss\_misc\_api.h, 759

VTSS\_OS\_TIMESTAMP  
     vtss\_misc\_api.h, 759

VTSS\_OS\_VIRT\_TO\_PHYS  
     vtss\_os\_ecos.h, 797

VTSS\_PACKET\_HDR\_SIZE\_BYTES  
     vtss\_packet\_api.h, 813

VTSS\_PACKET\_RATE\_DISABLED  
     types.h, 602

VTSS\_PACKET\_RX\_GRP\_CNT  
     types.h, 613

VTSS\_PACKET\_RX\_QUEUE\_CNT  
     types.h, 613

VTSS\_PACKET\_RX\_QUEUE\_END  
     types.h, 614

VTSS\_PACKET\_RX\_QUEUE\_NONE  
     types.h, 613

VTSS\_PACKET\_RX\_QUEUE\_START  
     types.h, 614

VTSS\_PACKET\_TX\_GRP\_CNT  
     types.h, 613

VTSS\_PACKET\_TX\_IFH\_MAX  
     vtss\_packet\_api.h, 814

VTSS\_PCP\_ARRAY\_SIZE  
     types.h, 606

VTSS\_PCP\_END  
     types.h, 606

VTSS\_PCP\_START  
     types.h, 606

VTSS\_PCPSP  
     types.h, 606

VTSS\_PHY\_10G\_FEC\_FIXED\_CNT\_THRESH\_EV  
     vtss\_phy\_10g\_api.h, 858

VTSS\_PHY\_10G\_FEC\_UNFIXED\_CNT\_THRESH\_EV  
     vtss\_phy\_10g\_api.h, 857

VTSS\_PHY\_10G\_FIFO\_SYNC  
     options.h, 574

VTSS\_PHY\_10G\_GPIO\_INT\_AGG0\_EV  
     vtss\_phy\_10g\_api.h, 858

VTSS\_PHY\_10G\_GPIO\_INT\_AGG1\_EV  
     vtss\_phy\_10g\_api.h, 859

VTSS\_PHY\_10G\_GPIO\_INT\_AGG2\_EV  
     vtss\_phy\_10g\_api.h, 859

VTSS\_PHY\_10G\_GPIO\_INT\_AGG3\_EV  
     vtss\_phy\_10g\_api.h, 859

VTSS\_PHY\_10G\_HIGH\_BER\_EV  
     vtss\_phy\_10g\_api.h, 850

VTSS\_PHY\_10G\_HIGHBER\_EV  
     vtss\_phy\_10g\_api.h, 858

VTSS\_PHY\_10G\_HOST\_MAC\_LOCAL\_FAULT\_EV  
     vtss\_phy\_10g\_api.h, 860

VTSS\_PHY\_10G\_HOST\_MAC\_REMOTE\_FAULT\_EV

vtss\_phy\_10g\_api.h, 860  
VTSS\_PHY\_10G\_LINE\_MAC\_LOCAL\_FAULT\_EV  
vtss\_phy\_10g\_api.h, 860  
VTSS\_PHY\_10G\_LINE\_MAC\_REMOTE\_FAULT\_EV  
vtss\_phy\_10g\_api.h, 860  
VTSS\_PHY\_10G\_LINK\_LOS\_EV  
vtss\_phy\_10g\_api.h, 849  
VTSS\_PHY\_10G\_LOPC\_EV  
vtss\_phy\_10g\_api.h, 850  
VTSS\_PHY\_10G\_MACSEC\_DISABLED  
vtss\_phy\_10g\_api.h, 848  
VTSS\_PHY\_10G\_MACSEC\_KEY\_128  
vtss\_phy\_10g\_api.h, 849  
VTSS\_PHY\_10G\_MODULE\_STAT\_EV  
vtss\_phy\_10g\_api.h, 850  
VTSS\_PHY\_10G\_ONE\_LINE\_ACTIVE  
vtss\_phy\_10g\_api.h, 848  
VTSS\_PHY\_10G\_PCS\_RECEIVE\_FAULT\_EV  
vtss\_phy\_10g\_api.h, 850  
VTSS\_PHY\_10G\_RX\_BLK\_DEC\_CNT\_THRESH\_EV  
vtss\_phy\_10g\_api.h, 857  
VTSS\_PHY\_10G\_RX\_CHAR\_DEC\_CNT\_THRESH\_EV  
vtss\_phy\_10g\_api.h, 856  
VTSS\_PHY\_10G\_RX\_LINK\_STAT\_EV  
vtss\_phy\_10g\_api.h, 858  
VTSS\_PHY\_10G\_RX\_LOL\_EV  
vtss\_phy\_10g\_api.h, 849, 856  
VTSS\_PHY\_10G\_RX\_LOS\_EV  
vtss\_phy\_10g\_api.h, 856  
VTSS\_PHY\_10G\_RX\_SEQ\_CNT\_THRESH\_EV  
vtss\_phy\_10g\_api.h, 857  
VTSS\_PHY\_10G\_TIMESTAMP\_DISABLED  
vtss\_phy\_10g\_api.h, 848  
VTSS\_PHY\_10G\_TX\_BLK\_ENC\_CNT\_THRESH\_EV  
vtss\_phy\_10g\_api.h, 857  
VTSS\_PHY\_10G\_TX\_CHAR\_ENC\_CNT\_THRESH\_EV  
vtss\_phy\_10g\_api.h, 856  
VTSS\_PHY\_10G\_TX\_LOL\_EV  
vtss\_phy\_10g\_api.h, 850, 856  
VTSS\_PHY\_10G\_TX\_SEQ\_CNT\_THRESH\_EV  
vtss\_phy\_10g\_api.h, 857  
VTSS\_PHY\_1G\_HOST\_AUTONEG\_RESTART\_EV  
vtss\_phy\_10g\_api.h, 859  
VTSS\_PHY\_1G\_LINE\_AUTONEG\_RESTART\_EV  
vtss\_phy\_10g\_api.h, 859  
VTSS\_PHY\_ACTIPHY\_PWR  
vtss\_phy\_api.h, 937  
VTSS\_PHY\_EWIS\_AISL\_EV  
vtss\_phy\_10g\_api.h, 852  
VTSS\_PHY\_EWIS\_AISP\_EV  
vtss\_phy\_10g\_api.h, 852  
VTSS\_PHY\_EWIS\_B1\_NZ\_EV  
vtss\_phy\_10g\_api.h, 854  
VTSS\_PHY\_EWIS\_B1\_THRESH\_EV  
vtss\_phy\_10g\_api.h, 855  
VTSS\_PHY\_EWIS\_B2\_NZ\_EV  
vtss\_phy\_10g\_api.h, 854  
VTSS\_PHY\_EWIS\_B2\_THRESH\_EV  
vtss\_phy\_10g\_api.h, 855  
VTSS\_PHY\_EWIS\_B3\_NZ\_EV  
vtss\_phy\_10g\_api.h, 854  
VTSS\_PHY\_EWIS\_B3\_THRESH\_EV  
vtss\_phy\_10g\_api.h, 855  
VTSS\_PHY\_EWIS\_FAIS\_EV  
vtss\_phy\_10g\_api.h, 851  
VTSS\_PHY\_EWIS\_FERDIP\_EV  
vtss\_phy\_10g\_api.h, 853  
VTSS\_PHY\_EWIS\_FEUNEQP\_EV  
vtss\_phy\_10g\_api.h, 853  
VTSS\_PHY\_EWIS\_FPLM\_EV  
vtss\_phy\_10g\_api.h, 851  
VTSS\_PHY\_EWIS\_LCDP\_EV  
vtss\_phy\_10g\_api.h, 852  
VTSS\_PHY\_EWIS\_LOF\_EV  
vtss\_phy\_10g\_api.h, 851  
VTSS\_PHY\_EWIS\_LOPP\_EV  
vtss\_phy\_10g\_api.h, 852  
VTSS\_PHY\_EWIS\_PLMP\_EV  
vtss\_phy\_10g\_api.h, 852  
VTSS\_PHY\_EWIS\_RDIL\_EV  
vtss\_phy\_10g\_api.h, 851  
VTSS\_PHY\_EWIS\_REIL\_EV  
vtss\_phy\_10g\_api.h, 853  
VTSS\_PHY\_EWIS\_REIL\_NZ\_EV  
vtss\_phy\_10g\_api.h, 854  
VTSS\_PHY\_EWIS\_REIL\_THRESH\_EV  
vtss\_phy\_10g\_api.h, 855  
VTSS\_PHY\_EWIS\_REIP\_EV  
vtss\_phy\_10g\_api.h, 853  
VTSS\_PHY\_EWIS\_REIP\_NZ\_EV  
vtss\_phy\_10g\_api.h, 854  
VTSS\_PHY\_EWIS\_REIP\_THRESH\_EV  
vtss\_phy\_10g\_api.h, 855  
VTSS\_PHY\_EWIS\_SEF\_EV  
vtss\_phy\_10g\_api.h, 851  
VTSS\_PHY\_EWIS\_UEQP\_EV  
vtss\_phy\_10g\_api.h, 853  
VTSS\_PHY\_LINK\_AMS\_EV  
vtss\_phy\_api.h, 942  
VTSS\_PHY\_LINK\_AUTO\_NEG\_COMPLETE\_EV  
vtss\_phy\_api.h, 942  
VTSS\_PHY\_LINK\_AUTO\_NEG\_ERROR\_EV  
vtss\_phy\_api.h, 942  
VTSS\_PHY\_LINK\_DOWN\_PWR  
vtss\_phy\_api.h, 937  
VTSS\_PHY\_LINK\_EXT\_EEE\_LINKFAIL\_EV  
vtss\_phy\_api.h, 945  
VTSS\_PHY\_LINK\_EXT\_EEE\_WAIT\_RX\_TQ\_EV  
vtss\_phy\_api.h, 945  
VTSS\_PHY\_LINK\_EXT\_EEE\_WAIT\_TS\_EV  
vtss\_phy\_api.h, 945  
VTSS\_PHY\_LINK\_EXT\_EEE\_WAKE\_ERR\_EV  
vtss\_phy\_api.h, 945  
VTSS\_PHY\_LINK\_EXT\_MACSEC\_EGRESS\_EV  
vtss\_phy\_api.h, 946

VTSS\_PHY\_LINK\_EXT\_MACSEC\_FC\_BUFF\_EV  
vtss\_phy\_api.h, 946

VTSS\_PHY\_LINK\_EXT\_MACSEC\_HOST\_MAC\_EV  
vtss\_phy\_api.h, 946

VTSS\_PHY\_LINK\_EXT\_MACSEC\_INGRESS\_EV  
vtss\_phy\_api.h, 946

VTSS\_PHY\_LINK\_EXT\_MACSEC\_LINE\_MAC\_EV  
vtss\_phy\_api.h, 946

VTSS\_PHY\_LINK\_EXT\_MEM\_INT\_RING\_EV  
vtss\_phy\_api.h, 947

VTSS\_PHY\_LINK\_EXT\_RR\_SW\_COMPL\_EV  
vtss\_phy\_api.h, 945

VTSS\_PHY\_LINK\_EXTENDED\_REG\_INT\_EV  
vtss\_phy\_api.h, 944

VTSS\_PHY\_LINK\_FALSE\_CARRIER\_INT\_EV  
vtss\_phy\_api.h, 943

VTSS\_PHY\_LINK\_FDX\_STATE\_CHANGE\_EV  
vtss\_phy\_api.h, 942

VTSS\_PHY\_LINK\_FFAIL\_EV  
vtss\_phy\_api.h, 941

VTSS\_PHY\_LINK\_INLINE\_POW\_DEV\_DETECT\_EV  
vtss\_phy\_api.h, 943

VTSS\_PHY\_LINK\_LINK\_SPEED\_DS\_DETECT\_EV  
vtss\_phy\_api.h, 944

VTSS\_PHY\_LINK\_LOS\_EV  
vtss\_phy\_api.h, 941

VTSS\_PHY\_LINK\_MASTER\_SLAVE\_RES\_ERR\_EV  
vtss\_phy\_api.h, 944

VTSS\_PHY\_LINK\_RX\_ER\_INT\_EV  
vtss\_phy\_api.h, 944

VTSS\_PHY\_LINK\_RX\_FIFO\_OVERFLOW\_INT\_EV  
vtss\_phy\_api.h, 943

VTSS\_PHY\_LINK\_SPEED\_STATE\_CHANGE\_EV  
vtss\_phy\_api.h, 942

VTSS\_PHY\_LINK\_SYMBOL\_ERR\_INT\_EV  
vtss\_phy\_api.h, 943

VTSS\_PHY\_LINK\_TX\_FIFO\_OVERFLOW\_INT\_EV  
vtss\_phy\_api.h, 943

VTSS\_PHY\_LINK\_UP\_FULL\_PWR  
vtss\_phy\_api.h, 937

VTSS\_PHY\_LINK\_WAKE\_ON\_LAN\_INT\_EV  
vtss\_phy\_api.h, 944

VTSS\_PHY\_OPT\_VERIPH  
options.h, 575

VTSS\_PHY\_PAGE\_0x2DAF  
vtss\_phy\_api.h, 940

VTSS\_PHY\_PAGE\_1588  
vtss\_phy\_api.h, 939

VTSS\_PHY\_PAGE\_EXTENDED\_2  
vtss\_phy\_api.h, 938

VTSS\_PHY\_PAGE\_EXTENDED\_3  
vtss\_phy\_api.h, 939

VTSS\_PHY\_PAGE\_EXTENDED\_4  
vtss\_phy\_api.h, 939

VTSS\_PHY\_PAGE\_EXTENDED  
vtss\_phy\_api.h, 938

VTSS\_PHY\_PAGE\_GPIO  
vtss\_phy\_api.h, 939

VTSS\_PHY\_PAGE\_MACSEC  
vtss\_phy\_api.h, 939

VTSS\_PHY\_PAGE\_STANDARD  
vtss\_phy\_api.h, 938

VTSS\_PHY\_PAGE\_TEST  
vtss\_phy\_api.h, 940

VTSS\_PHY\_PAGE\_TR  
vtss\_phy\_api.h, 940

VTSS\_PHY\_POWER\_ACTIPHY\_BIT  
phy.h, 578  
vtss\_phy\_api.h, 936

VTSS\_PHY\_POWER\_DYNAMIC\_BIT  
phy.h, 578  
vtss\_phy\_api.h, 937

VTSS\_PHY\_RECov\_CLK1  
vtss\_phy\_api.h, 937

VTSS\_PHY\_RECov\_CLK2  
vtss\_phy\_api.h, 938

VTSS\_PHY\_RECov\_CLK\_NUM  
vtss\_phy\_api.h, 938

VTSS\_PHY\_REG\_EXTENDED  
vtss\_phy\_api.h, 940

VTSS\_PHY\_REG\_GPIO  
vtss\_phy\_api.h, 941

VTSS\_PHY\_REG\_STANDARD  
vtss\_phy\_api.h, 940

VTSS\_PHY\_REG\_TEST  
vtss\_phy\_api.h, 941

VTSS\_PHY\_REG\_TR  
vtss\_phy\_api.h, 941

VTSS\_PHY\_TS\_8487\_XAUI\_SEL\_0  
vtss\_phy\_ts\_api.h, 1023

VTSS\_PHY\_TS\_8487\_XAUI\_SEL\_1  
vtss\_phy\_ts\_api.h, 1024

VTSS\_PHY\_TS\_DATA\_IN\_RSRVD\_FIELD  
vtss\_phy\_ts\_api.h, 1023

VTSS\_PHY\_TS\_EGR\_DATAPATH\_RESET  
vtss\_phy\_ts\_api.h, 1024

VTSS\_PHY\_TS\_EGR\_ENGINE\_ERR  
vtss\_phy\_ts\_api.h, 1022

VTSS\_PHY\_TS\_EGR\_FIFO\_OVERFLOW  
vtss\_phy\_ts\_api.h, 1023

VTSS\_PHY\_TS\_EGR\_FIFO\_RESET  
vtss\_phy\_ts\_api.h, 1025

VTSS\_PHY\_TS\_EGR\_LTC2\_RESET  
vtss\_phy\_ts\_api.h, 1024

VTSS\_PHY\_TS\_EGR\_RW\_FCS\_ERR  
vtss\_phy\_ts\_api.h, 1022

VTSS\_PHY\_TS\_EGR\_TIMESTAMP\_CAPTURED  
vtss\_phy\_ts\_api.h, 1022

VTSS\_PHY\_TS\_ENG\_FLOW\_VALID\_FOR\_CH0  
vtss\_phy\_ts\_api.h, 1021

VTSS\_PHY\_TS\_ENG\_FLOW\_VALID\_FOR\_CH1  
vtss\_phy\_ts\_api.h, 1021

VTSS\_PHY\_TS\_ETH\_ADDR\_MATCH\_48BIT  
vtss\_phy\_ts\_api.h, 1016

VTSS\_PHY\_TS\_ETH\_ADDR\_MATCH\_ANY\_MULTI-  
CAST

vtss\_phy\_ts\_api.h, 1017  
VTSS\_PHY\_TS\_ETH\_ADDR\_MATCH\_ANY\_UNICAST  
    vtss\_phy\_ts\_api.h, 1016  
VTSS\_PHY\_TS\_ETH\_MATCH\_DEST\_ADDR  
    vtss\_phy\_ts\_api.h, 1017  
VTSS\_PHY\_TS\_ETH\_MATCH\_SRC\_ADDR  
    vtss\_phy\_ts\_api.h, 1017  
VTSS\_PHY\_TS\_ETH\_MATCH\_SRC\_OR\_DEST  
    vtss\_phy\_ts\_api.h, 1017  
VTSS\_PHY\_TS\_FIFO\_SIG\_DEST\_IP  
    vtss\_phy\_ts\_api.h, 1013  
VTSS\_PHY\_TS\_FIFO\_SIG\_DEST\_MAC  
    vtss\_phy\_ts\_api.h, 1014  
VTSS\_PHY\_TS\_FIFO\_SIG\_DOMAIN\_NUM  
    vtss\_phy\_ts\_api.h, 1013  
VTSS\_PHY\_TS\_FIFO\_SIG\_MSG\_TYPE  
    vtss\_phy\_ts\_api.h, 1013  
VTSS\_PHY\_TS\_FIFO\_SIG\_SEQ\_ID  
    vtss\_phy\_ts\_api.h, 1014  
VTSS\_PHY\_TS\_FIFO\_SIG\_SOURCE\_PORT\_ID  
    vtss\_phy\_ts\_api.h, 1013  
VTSS\_PHY\_TS\_FIFO\_SIG\_SRC\_IP  
    vtss\_phy\_ts\_api.h, 1013  
VTSS\_PHY\_TS\_INGR\_DATAPATH\_RESET  
    vtss\_phy\_ts\_api.h, 1024  
VTSS\_PHY\_TS\_INGR\_ENGINE\_ERR  
    vtss\_phy\_ts\_api.h, 1021  
VTSS\_PHY\_TS\_INGR\_LTC1\_RESET  
    vtss\_phy\_ts\_api.h, 1024  
VTSS\_PHY\_TS\_INGR\_RW\_FCS\_ERR  
    vtss\_phy\_ts\_api.h, 1022  
VTSS\_PHY\_TS\_INGR\_RW\_PREAM\_ERR  
    vtss\_phy\_ts\_api.h, 1022  
VTSS\_PHY\_TS\_IP\_MATCH\_DEST  
    vtss\_phy\_ts\_api.h, 1019  
VTSS\_PHY\_TS\_IP\_MATCH\_SRC\_OR\_DEST  
    vtss\_phy\_ts\_api.h, 1020  
VTSS\_PHY\_TS\_IP\_MATCH\_SRC  
    vtss\_phy\_ts\_api.h, 1019  
VTSS\_PHY\_TS\_IP\_VER\_4  
    vtss\_phy\_ts\_api.h, 1019  
VTSS\_PHY\_TS\_IP\_VER\_6  
    vtss\_phy\_ts\_api.h, 1019  
VTSS\_PHY\_TS\_LTC\_LOAD\_SAVE\_NEW\_TOD  
    vtss\_phy\_ts\_api.h, 1023  
VTSS\_PHY\_TS\_LTC\_NEW\_PPS\_INTRPT  
    vtss\_phy\_ts\_api.h, 1023  
VTSS\_PHY\_TS\_MPLS\_STACK\_DEPTH\_1  
    vtss\_phy\_ts\_api.h, 1020  
VTSS\_PHY\_TS\_MPLS\_STACK\_DEPTH\_2  
    vtss\_phy\_ts\_api.h, 1020  
VTSS\_PHY\_TS\_MPLS\_STACK\_DEPTH\_3  
    vtss\_phy\_ts\_api.h, 1020  
VTSS\_PHY\_TS\_MPLS\_STACK\_DEPTH\_4  
    vtss\_phy\_ts\_api.h, 1020  
VTSS\_PHY\_TS\_MPLS\_STACK\_REF\_POINT\_END  
    vtss\_phy\_ts\_api.h, 1021  
VTSS\_PHY\_TS\_MPLS\_STACK\_REF\_POINT\_TOP  
    vtss\_phy\_ts\_api.h, 1021  
vtss\_phy\_ts\_api.h, 1021  
VTSS\_PHY\_TS\_SIG\_DEST\_IP\_LEN  
    vtss\_phy\_ts\_api.h, 1015  
VTSS\_PHY\_TS\_SIG\_DEST\_MAC\_LEN  
    vtss\_phy\_ts\_api.h, 1016  
VTSS\_PHY\_TS\_SIG\_DOMAIN\_NUM\_LEN  
    vtss\_phy\_ts\_api.h, 1014  
VTSS\_PHY\_TS\_SIG\_LEN  
    vtss\_phy\_ts\_api.h, 1014  
VTSS\_PHY\_TS\_SIG\_MSG\_TYPE\_LEN  
    vtss\_phy\_ts\_api.h, 1015  
VTSS\_PHY\_TS\_SIG\_SEQ\_ID\_LEN  
    vtss\_phy\_ts\_api.h, 1015  
VTSS\_PHY\_TS\_SIG\_SEQUENCE\_ID\_LEN  
    vtss\_phy\_ts\_api.h, 1015  
VTSS\_PHY\_TS\_SIG\_SOURCE\_PORT\_ID\_LEN  
    vtss\_phy\_ts\_api.h, 1015  
VTSS\_PHY\_TS\_SIG\_SRC\_IP\_LEN  
    vtss\_phy\_ts\_api.h, 1016  
VTSS\_PHY\_TS\_SIG\_TIME\_STAMP\_LEN  
    vtss\_phy\_ts\_api.h, 1014  
VTSS\_PHY\_TS\_SPI\_CLK\_THRU\_PPS0  
    options.h, 561  
VTSS\_PHY\_TS\_TAG\_RANGE\_INNER  
    vtss\_phy\_ts\_api.h, 1019  
VTSS\_PHY\_TS\_TAG\_RANGE\_NONE  
    vtss\_phy\_ts\_api.h, 1018  
VTSS\_PHY\_TS\_TAG\_RANGE\_OUTER  
    vtss\_phy\_ts\_api.h, 1018  
VTSS\_PHY\_TS\_TAG\_TYPE\_B  
    vtss\_phy\_ts\_api.h, 1018  
VTSS\_PHY\_TS\_TAG\_TYPE\_C  
    vtss\_phy\_ts\_api.h, 1017  
VTSS\_PHY\_TS\_TAG\_TYPE\_I  
    vtss\_phy\_ts\_api.h, 1018  
VTSS\_PHY\_TS\_TAG\_TYPE\_S  
    vtss\_phy\_ts\_api.h, 1018  
VTSS\_PHYS\_PORT\_CNT  
    vtss\_fdma\_api.h, 644  
VTSS\_PORT\_ARRAY\_SIZE  
    types.h, 603  
VTSS\_PORT\_COUNT  
    types.h, 602  
VTSS\_PORT\_IS\_PORT  
    types.h, 604  
VTSS\_PORT\_NO\_ANY  
    vtss\_security\_api.h, 1102  
VTSS\_PORT\_NO\_CPU  
    types.h, 603  
VTSS\_PORT\_NO\_END  
    types.h, 603  
VTSS\_PORT\_NO\_NONE  
    types.h, 603  
VTSS\_PORT\_NO\_START  
    types.h, 603  
VTSS\_PORT\_POLICER\_CPU\_QUEUES  
    vtss\_qos\_api.h, 1094  
VTSS\_PORT\_POLICERS

vtss\_qos\_api.h, 1093  
**VTSS\_PORTS**  
 types.h, 602  
**VTSS\_PRIO\_ARRAY\_SIZE**  
 types.h, 605  
**VTSS\_PRIO\_END**  
 types.h, 604  
**VTSS\_PRIO\_NO\_NONE**  
 types.h, 604  
**VTSS\_PRIO\_START**  
 types.h, 604  
**VTSS\_PRIO\_SUPER**  
 vtss\_packet\_api.h, 812  
**VTSS\_PRIOS**  
 types.h, 604  
**VTSS\_PTP\_IP\_1588\_VERSION\_2\_1**  
 vtss\_phy\_ts\_api.h, 1016  
**VTSS\_PVLAN\_ARRAY\_SIZE**  
 vtss\_l2\_api.h, 691  
**VTSS\_PVLAN\_NO\_DEFAULT**  
 vtss\_l2\_api.h, 691  
**VTSS\_PVLAN\_NO\_END**  
 vtss\_l2\_api.h, 691  
**VTSS\_PVLAN\_NO\_START**  
 vtss\_l2\_api.h, 690  
**VTSS\_PVLANS**  
 vtss\_l2\_api.h, 690  
**VTSS\_QCE\_ID\_LAST**  
 vtss\_qos\_api.h, 1095  
**VTSS\_QCL\_ARRAY\_SIZE**  
 vtss\_qos\_api.h, 1094  
**VTSS\_QCL\_ID\_END**  
 vtss\_qos\_api.h, 1094  
**VTSS\_QCL\_ID\_START**  
 vtss\_qos\_api.h, 1094  
**VTSS\_QCL\_IDS**  
 vtss\_qos\_api.h, 1094  
**VTSS\_QS\_CONF\_MAX**  
 vtss\_init\_api.h, 662  
**VTSS\_QS\_CONF\_MIN**  
 vtss\_init\_api.h, 663  
**VTSS\_QUEUE\_ARRAY\_SIZE**  
 types.h, 605  
**VTSS\_QUEUE\_END**  
 types.h, 605  
**VTSS\_QUEUE\_START**  
 types.h, 605  
**VTSS\_QUEUES**  
 types.h, 605  
**VTSS\_RLEG\_CNT**  
 vtss\_l3\_api.h, 744  
**VTSS\_SEC\_NS\_INTERVAL**  
 types.h, 618  
**VTSS\_SLEWRATE\_120PS**  
 vtss\_phy\_10g\_api.h, 847  
**VTSS\_SLEWRATE\_25PS**  
 vtss\_phy\_10g\_api.h, 846  
**VTSS\_SLEWRATE\_35PS**  
 vtss\_phy\_10g\_api.h, 846  
**VTSS\_SLEWRATE\_55PS**  
 vtss\_phy\_10g\_api.h, 846  
**VTSS\_SLEWRATE\_70PS**  
 vtss\_phy\_10g\_api.h, 847  
**VTSS\_SLEWRATE\_INVALID**  
 vtss\_phy\_10g\_api.h, 847  
**VTSS\_SVL\_PACKET\_HDR\_SIZE\_BYTES**  
 vtss\_packet\_api.h, 813  
**VTSS\_SVL\_RX\_IFH\_SIZE**  
 vtss\_packet\_api.h, 813  
**VTSS\_SYNCE\_CLK\_MAX**  
 vtss\_sync\_api.h, 1111  
**VTSS\_SYNCE\_CLK\_PORT\_ARRAY\_SIZE**  
 types.h, 618  
**VTSS\_SYNCE\_CLK\_A**  
 vtss\_sync\_api.h, 1110  
**VTSS\_SYNCE\_CLK\_B**  
 vtss\_sync\_api.h, 1110  
**VTSS\_TIME\_OF\_DAY**  
 vtss\_os\_ecos.h, 793  
 vtss\_os\_linux.h, 804  
**VTSS\_UPSPN\_CPU**  
 vtss\_l2\_api.h, 686  
**VTSS\_UPSPN\_NONE**  
 vtss\_l2\_api.h, 686  
**VTSS\_VCE\_ID\_LAST**  
 vtss\_l2\_api.h, 688  
**VTSS\_VCL\_ARRAY\_SIZE**  
 vtss\_l2\_api.h, 687  
**VTSS\_VCL\_ID\_END**  
 vtss\_l2\_api.h, 687  
**VTSS\_VCL\_ID\_START**  
 vtss\_l2\_api.h, 687  
**VTSS\_VCL\_IDS**  
 vtss\_l2\_api.h, 687  
**VTSS\_VID\_ALL**  
 types.h, 609  
**VTSS\_VID\_DEFAULT**  
 types.h, 609  
**VTSS\_VID\_NULL**  
 types.h, 608  
**VTSS\_VID\_RESERVED**  
 types.h, 609  
**VTSS\_VIDS**  
 types.h, 609  
**VTSS\_VLAN\_TRANS\_FIRST\_GROUP\_ID**  
 vtss\_l2\_api.h, 688  
**VTSS\_VLAN\_TRANS\_GROUP\_MAX\_CNT**  
 vtss\_l2\_api.h, 688  
**VTSS\_VLAN\_TRANS\_LAST\_GROUP\_ID**  
 vtss\_l2\_api.h, 689  
**VTSS\_VLAN\_TRANS\_MAX\_CNT**  
 vtss\_l2\_api.h, 688  
**VTSS\_VLAN\_TRANS\_MAX\_VLAN\_ID**  
 vtss\_l2\_api.h, 689  
**VTSS\_VLAN\_TRANS\_NULL\_CHECK**  
 vtss\_l2\_api.h, 690

VTSS\_VLAN\_TRANS\_NULL\_GROUP\_ID  
vtss\_l2\_api.h, 688

VTSS\_VLAN\_TRANS\_PORT\_BF\_SIZE  
vtss\_l2\_api.h, 690

VTSS\_VLAN\_TRANS\_VALID\_GROUP\_CHECK  
vtss\_l2\_api.h, 689

VTSS\_VLAN\_TRANS\_VALID\_VLAN\_CHECK  
vtss\_l2\_api.h, 689

VTSS\_VLAN\_TRANS\_VID\_START  
vtss\_l2\_api.h, 689

VTSS\_VSTAX\_HDR\_SIZE  
vtss\_packet\_api.h, 812

VTSS\_VSTAX\_TTL\_PORT  
vtss\_packet\_api.h, 812

VTSS\_VSTAX\_UPSID\_LEGAL  
vtss\_l2\_api.h, 685

VTSS\_VSTAX\_UPSID\_MAX  
vtss\_l2\_api.h, 685

VTSS\_VSTAX\_UPSID\_MIN  
vtss\_l2\_api.h, 685

VTSS\_VSTAX\_UPSID\_START  
vtss\_l2\_api.h, 685

VTSS\_VSTAX\_UPSID\_UNDEF  
vtss\_l2\_api.h, 685

VTSS\_VSTAX\_UPSIDS  
vtss\_l2\_api.h, 684

val  
vtss\_eee\_port\_state\_t, 93  
vtss\_phy\_conf\_1g\_t, 334  
vtss\_phy\_ts\_eth\_conf\_t, 375, 376  
vtss\_phy\_ts\_ptp\_conf\_t, 405  
vtss\_phy\_ts\_y1731\_oam\_conf\_t, 414

valid  
vtss\_ewis\_tti\_s, 132  
vtss\_vstax\_rx\_header\_t, 551

value  
ib\_par\_cfg, 31  
vtss\_phy\_ts\_ach\_conf\_t, 363  
vtss\_phy\_ts\_eth\_conf\_t, 375, 376  
vtss\_phy\_ts\_ptp\_conf\_t, 406  
vtss\_phy\_ts\_y1731\_oam\_conf\_t, 415  
vtss\_sgpi\_port\_data\_t, 496  
vtss\_vcap\_ip\_t, 511  
vtss\_vcap\_u128\_t, 512  
vtss\_vcap\_u16\_t, 513  
vtss\_vcap\_u24\_t, 514  
vtss\_vcap\_u32\_t, 515  
vtss\_vcap\_u40\_t, 516  
vtss\_vcap\_u48\_t, 517  
vtss\_vcap\_u8\_t, 518  
vtss\_vcap\_vid\_t, 520  
vtss\_vcap\_vr\_t, 522

venice\_rev\_a\_los\_detection\_workaround  
vtss\_phy\_10g\_mode\_t, 289

version  
vtss\_phy\_ts\_ach\_conf\_t, 363  
vtss\_phy\_ts\_oam\_engine\_action\_t, 401

vga\_adc\_debug

vtss\_phy\_api.h, 976

vid  
vtss\_ace\_vlan\_t, 59  
vtss\_ece\_inner\_tag\_t, 81  
vtss\_ece\_outer\_tag\_t, 85  
vtss\_ece\_tag\_t, 88  
vtss\_evc\_pb\_conf\_t, 98  
vtss\_mirror\_conf\_t, 189  
vtss\_packet\_frame\_info\_t, 194  
vtss\_packet\_port\_info\_t, 197  
vtss\_qce\_tag\_t, 473  
vtss\_tci\_t, 501  
vtss\_vce\_action\_t, 523  
vtss\_vce\_tag\_t, 535  
vtss\_vid\_mac\_t, 537  
vtss\_vlan\_tag\_t, 541  
vtss\_vlan\_trans\_grp2vlan\_conf\_t, 542

vid\_mac  
vtss\_mac\_table\_entry\_t, 184

vlan  
vtss\_ace\_t, 57  
vtss\_l3\_neighbour\_t, 177  
vtss\_l3\_rleg\_conf\_t, 179  
vtss\_routing\_entry\_t, 490

vlan\_check  
vtss\_phy\_ts\_eth\_conf\_t, 373

vml\_format  
vtss\_debug\_info\_t, 72

vp  
vtss\_phy\_10g\_ob\_status\_t, 294

vr  
vtss\_vcap\_vr\_t, 523

vrid0  
vtss\_l3\_rleg\_conf\_t, 179

vrid0\_enable  
vtss\_l3\_rleg\_conf\_t, 179

vrid1  
vtss\_l3\_rleg\_conf\_t, 180

vrid1\_enable  
vtss\_l3\_rleg\_conf\_t, 180

vstax  
vtss\_packet\_rx\_header\_t, 201  
vtss\_packet\_rx\_info\_t, 210  
vtss\_packet\_tx\_info\_t, 229

vstax2  
vtss\_mac\_table\_entry\_t, 186

vtail  
vtss\_phy\_10g\_jitter\_conf\_t, 271

vtss\_10g\_phy\_gpio\_t  
vtss\_phy\_10g\_api.h, 876

vtss\_32\_cntr\_t  
vtss\_phy\_10g\_api.h, 861

vtss\_ace\_add  
vtss\_security\_api.h, 1107

vtss\_ace\_bit\_t  
vtss\_security\_api.h, 1103

vtss\_ace\_counter\_clear  
vtss\_security\_api.h, 1109

vtss\_ace\_counter\_get  
     vtss\_security\_api.h, 1108  
 vtss\_ace\_del  
     vtss\_security\_api.h, 1108  
 vtss\_ace\_frame\_arp\_t, 40  
     arp, 40  
     dip, 42  
     dmac\_match, 41  
     ethernet, 42  
     ip, 41  
     length, 41  
     req, 40  
     sip, 42  
     smac, 40  
     smac\_match, 41  
     unknown, 41  
 vtss\_ace\_frame\_etype\_t, 42  
     data, 43  
     dmac, 43  
     etype, 43  
     smac, 43  
 vtss\_ace\_frame\_ipv4\_t, 44  
     data, 46  
     dip, 46  
     dport, 46  
     ds, 45  
     fragment, 45  
     options, 45  
     proto, 45  
     seq\_zero, 48  
     sip, 45  
     sip\_eq\_dip, 48  
     sport, 46  
     sport\_eq\_dport, 48  
     tcp\_ack, 47  
     tcp\_fin, 46  
     tcp\_psh, 47  
     tcp\_RST, 47  
     tcp\_SYN, 47  
     tcp\_URG, 47  
     ttl, 44  
 vtss\_ace\_frame\_ipv6\_t, 48  
     data, 50  
     dport, 50  
     ds, 50  
     proto, 49  
     seq\_zero, 52  
     sip, 49  
     sip\_eq\_dip, 52  
     sport, 50  
     sport\_eq\_dport, 52  
     tcp\_ack, 51  
     tcp\_fin, 50  
     tcp\_psh, 51  
     tcp\_RST, 51  
     tcp\_SYN, 51  
     tcp\_URG, 51  
     ttl, 49

vtss\_ace\_frame\_llc\_t, 52  
     dmac, 53  
     llc, 53  
     smac, 53  
 vtss\_ace\_frame\_snap\_t, 54  
     dmac, 54  
     smac, 54  
     snap, 54

vtss\_ace\_init  
     vtss\_security\_api.h, 1107

vtss\_ace\_t, 55  
     action, 56  
     arp, 58  
     dmac\_bc, 57  
     dmac\_mc, 56  
     etype, 57  
     frame, 58  
     id, 55  
     ipv4, 58  
     ipv6, 58  
     llc, 57  
     policy, 56  
     port\_no, 56  
     snap, 57  
     type, 56  
     vlan, 57

vtss\_ace\_type\_t  
     vtss\_security\_api.h, 1103

vtss\_ace\_vlan\_t, 59  
     cfi, 59  
     usr\_prio, 59  
     vid, 59

vtss\_acl\_action\_t, 60  
     cpu, 60  
     cpu\_once, 60  
     cpu\_queue, 61  
     forward, 61  
     irq\_trigger, 62  
     learn, 61  
     police, 61  
     policer\_no, 61  
     port\_forward, 62  
     port\_no, 62

vtss\_acl\_policer\_conf\_get  
     vtss\_security\_api.h, 1104

vtss\_acl\_policer\_conf\_set  
     vtss\_security\_api.h, 1105

vtss\_acl\_policer\_conf\_t, 62  
     rate, 63

vtss\_acl\_port\_conf\_get  
     vtss\_security\_api.h, 1105

vtss\_acl\_port\_conf\_set  
     vtss\_security\_api.h, 1106

vtss\_acl\_port\_conf\_t, 63  
     action, 64  
     policy\_no, 64

vtss\_acl\_port\_counter\_clear  
     vtss\_security\_api.h, 1106

vtss\_acl\_port\_counter\_get  
    vtss\_security\_api.h, 1106  
vtss\_aggr\_glag\_members\_get  
    vtss\_l2\_api.h, 724  
vtss\_aggr\_mode\_get  
    vtss\_l2\_api.h, 724  
vtss\_aggr\_mode\_set  
    vtss\_l2\_api.h, 724  
vtss\_aggr\_mode\_t, 64  
    dmac\_enable, 65  
    sip\_dip\_enable, 65  
    smac\_enable, 65  
    sport\_dport\_enable, 65  
vtss\_aggr\_port\_members\_get  
    vtss\_l2\_api.h, 723  
vtss\_aggr\_port\_members\_set  
    vtss\_l2\_api.h, 723  
vtss\_aneg\_t, 66  
    generate\_pause, 66  
    obey\_pause, 66  
vtss\_api/include/vtss/api/l2\_types.h, 557  
vtss\_api/include/vtss/api/options.h, 558  
vtss\_api/include/vtss/api/phy.h, 578  
vtss\_api/include/vtss/api/port.h, 580  
vtss\_api/include/vtss/api/types.h, 592  
vtss\_api/include/vtss\_ae\_api.h, 630  
vtss\_api/include/vtss\_afi\_api.h, 630  
vtss\_api/include/vtss\_aneg\_api.h, 631  
vtss\_api/include/vtss\_api.h, 631  
vtss\_api/include/vtss\_evc\_api.h, 631  
vtss\_api/include/vtss\_fdma\_api.h, 642  
vtss\_api/include/vtss\_gfp\_api.h, 659  
vtss\_api/include/vtss\_hqos\_api.h, 659  
vtss\_api/include/vtss\_i2c\_api.h, 659  
vtss\_api/include/vtss\_init\_api.h, 660  
vtss\_api/include/vtss\_l2\_api.h, 676  
vtss\_api/include/vtss\_l3\_api.h, 741  
vtss\_api/include/vtss\_mac10g\_api.h, 753  
vtss\_api/include/vtss\_misc\_api.h, 753  
vtss\_api/include/vtss\_mpls\_api.h, 785  
vtss\_api/include/vtss\_oam\_api.h, 785  
vtss\_api/include/vtss\_oha\_api.h, 786  
vtss\_api/include/vtss\_os.h, 786  
vtss\_api/include/vtss\_os\_custom.h, 786  
vtss\_api/include/vtss\_os\_ecos.h, 791  
vtss\_api/include/vtss\_os\_linux.h, 801  
vtss\_api/include/vtss\_othn\_api.h, 808  
vtss\_api/include/vtss\_packet\_api.h, 808  
vtss\_api/include/vtss\_pcs\_10gbase\_r\_api.h, 831  
vtss\_api/include/vtss\_phy\_10g\_api.h, 831  
vtss\_api/include/vtss\_phy\_api.h, 926  
vtss\_api/include/vtss\_phy\_ts\_api.h, 1003  
vtss\_api/include/vtss\_port\_api.h, 1075  
vtss\_api/include/vtss\_qos\_api.h, 1091  
vtss\_api/include/vtss\_rab\_api.h, 1099  
vtss\_api/include/vtss\_security\_api.h, 1100  
vtss\_api/include/vtss\_sf4\_api.h, 1109  
vtss\_api/include/vtss\_sync\_api.h, 1109  
vtss\_api/include/vtss\_tfi5\_api.h, 1113  
vtss\_api/include/vtss\_ts\_api.h, 1113  
vtss\_api/include/vtss\_upi\_api.h, 1131  
vtss\_api/include/vtss\_wis\_api.h, 1131  
vtss\_api/include/vtss\_xaui\_api.h, 1163  
vtss\_api/include/vtss\_xfi\_api.h, 1163  
vtss\_api\_lock\_t, 66  
    file, 67  
    function, 67  
    inst, 67  
    line, 67  
vtss\_apvlan\_port\_members\_get  
    vtss\_l2\_api.h, 720  
vtss\_apvlan\_port\_members\_set  
    vtss\_l2\_api.h, 720  
vtss\_auth\_port\_state\_get  
    vtss\_security\_api.h, 1103  
vtss\_auth\_port\_state\_set  
    vtss\_security\_api.h, 1104  
vtss\_auth\_state\_t  
    vtss\_security\_api.h, 1102  
vtss\_basic\_counters\_t, 68  
    rx\_frames, 68  
    tx\_frames, 68  
vtss\_callout\_free  
    vtss\_os\_ecos.h, 800  
vtss\_callout\_lock  
    vtss\_misc\_api.h, 767  
vtss\_callout\_malloc  
    vtss\_os\_ecos.h, 800  
vtss\_callout\_trace\_hex\_dump  
    vtss\_misc\_api.h, 766  
vtss\_callout\_trace\_printf  
    vtss\_misc\_api.h, 765  
vtss\_callout\_unlock  
    vtss\_misc\_api.h, 767  
vtss\_channel\_t  
    vtss\_phy\_10g\_api.h, 867  
vtss\_chip\_id\_get  
    vtss\_misc\_api.h, 770  
vtss\_chip\_id\_t, 69  
    part\_number, 69  
    revision, 69  
vtss\_ckout\_data\_sel\_t  
    vtss\_phy\_10g\_api.h, 869  
vtss\_counter\_pair\_t, 70  
    bytes, 70  
    frames, 70  
vtss\_debug\_group\_t  
    vtss\_misc\_api.h, 761  
vtss\_debug\_info\_get  
    vtss\_misc\_api.h, 766  
vtss\_debug\_info\_print  
    vtss\_misc\_api.h, 767  
vtss\_debug\_info\_t, 71  
    chip\_no, 72  
    clear, 72  
    full, 72

group, 71  
 layer, 71  
 port\_list, 72  
 vml\_format, 72  
**vtss\_debug\_layer\_t**  
 vtss\_misc\_api.h, 761  
**vtss\_debug\_lock**  
 vtss\_misc\_api.h, 768  
**vtss\_debug\_lock\_t**, 73  
 chip\_no, 73  
**vtss\_debug\_reg\_check\_set**  
 vtss\_misc\_api.h, 784  
**vtss\_debug\_unlock**  
 vtss\_misc\_api.h, 768  
**vtss\_dev\_all\_event\_enable**  
 vtss\_misc\_api.h, 772  
**vtss\_dev\_all\_event\_poll**  
 vtss\_misc\_api.h, 772  
**vtss\_dgroup\_port\_conf\_get**  
 vtss\_l2\_api.h, 720  
**vtss\_dgroup\_port\_conf\_set**  
 vtss\_l2\_api.h, 721  
**vtss\_dgroup\_port\_conf\_t**, 74  
 dgroup\_no, 74  
**vtss\_dlb\_policer\_conf\_t**, 74  
 cbs, 76  
 cf, 75  
 cir, 76  
 cm, 75  
 ebs, 76  
 eir, 76  
 enable, 75  
 line\_rate, 75  
 type, 75  
**vtss\_dscp\_emode\_t**  
 vtss\_qos\_api.h, 1096  
**vtss\_dscp\_mode\_t**  
 vtss\_qos\_api.h, 1095  
**vtss\_ece\_action\_t**, 77  
 dir, 77  
 evc\_id, 78  
 inner\_tag, 78  
 outer\_tag, 77  
 policer\_id, 78  
 policy\_no, 78  
 pop\_tag, 77  
**vtss\_ece\_add**  
 vtss\_evc\_api.h, 639  
**vtss\_ece\_counters\_clear**  
 vtss\_evc\_api.h, 641  
**vtss\_ece\_counters\_get**  
 vtss\_evc\_api.h, 641  
**vtss\_ece\_del**  
 vtss\_evc\_api.h, 639  
**vtss\_ece\_dir\_t**  
 types.h, 628  
**vtss\_ece\_frame\_ipv4\_t**, 79  
 dscp, 79  
**vtss\_ece\_frame\_ipv6\_t**, 79  
 dscp, 80  
**vtss\_ece\_init**  
 vtss\_evc\_api.h, 638  
**vtss\_ece\_inner\_tag\_t**, 80  
 dei, 81  
 pcp, 81  
 pcp\_dei\_preserve, 81  
 type, 80  
 vid, 81  
**vtss\_ece\_inner\_tag\_type\_t**  
 types.h, 629  
**vtss\_ece\_key\_t**, 82  
 frame, 83  
 inner\_tag, 83  
 ipv4, 83  
 ipv6, 83  
 mac, 82  
 port\_list, 82  
 tag, 82  
 type, 83  
**vtss\_ece\_mac\_t**, 84  
 dmac, 84  
**vtss\_ece\_outer\_tag\_t**, 85  
 dei, 86  
 enable, 85  
 pcp, 86  
 pcp\_dei\_preserve, 85  
 vid, 85  
**vtss\_ece\_pop\_tag\_t**  
 types.h, 629  
**vtss\_ece\_port\_t**  
 vtss\_evc\_api.h, 635  
**vtss\_ece\_t**, 86  
 action, 87  
 id, 87  
 key, 87  
**vtss\_ece\_tag\_t**, 87  
 dei, 88  
 pcp, 88  
 s\_tagged, 88  
 tagged, 88  
 vid, 88  
**vtss\_ece\_type\_t**  
 vtss\_evc\_api.h, 635  
**vtss\_eee\_port\_conf\_set**  
 vtss\_misc\_api.h, 783  
**vtss\_eee\_port\_conf\_t**, 89  
 eee\_ena, 89  
 eee\_fast\_queues, 89  
 lp\_advertisement, 90  
 optimized\_for\_power, 90  
 tx\_tw, 90  
**vtss\_eee\_port\_counter\_get**  
 vtss\_misc\_api.h, 784  
**vtss\_eee\_port\_counter\_t**, 90  
 fill\_level, 91  
 fill\_level\_get, 91

fill\_level\_thres, 91  
tx\_out\_bytes, 92  
tx\_out\_bytes\_get, 91  
vtss\_eee\_port\_state\_set  
  vtss\_misc\_api.h, 783  
vtss\_eee\_port\_state\_t, 92  
  select, 92  
  val, 93  
vtss\_eee\_state\_select\_t  
  vtss\_misc\_api.h, 764  
vtss\_eps\_port\_conf\_get  
  vtss\_l2\_api.h, 735  
vtss\_eps\_port\_conf\_set  
  vtss\_l2\_api.h, 735  
vtss\_eps\_port\_conf\_t, 93  
  port\_no, 94  
  type, 93  
vtss\_eps\_port\_selector\_get  
  vtss\_l2\_api.h, 736  
vtss\_eps\_port\_selector\_set  
  vtss\_l2\_api.h, 736  
vtss\_eps\_port\_type\_t  
  vtss\_l2\_api.h, 694  
vtss\_eps\_selector\_t  
  vtss\_l2\_api.h, 694  
vtss\_erps\_port\_state\_get  
  vtss\_l2\_api.h, 738  
vtss\_erps\_port\_state\_set  
  vtss\_l2\_api.h, 738  
vtss\_erps\_state\_t  
  vtss\_l2\_api.h, 695  
vtss\_erps\_vlan\_member\_get  
  vtss\_l2\_api.h, 737  
vtss\_erps\_vlan\_member\_set  
  vtss\_l2\_api.h, 737  
vtss\_evc\_add  
  vtss\_evc\_api.h, 637  
vtss\_evc\_api.h  
  VTSS\_ECE\_ID\_LAST, 634  
  VTSS\_EVC\_ID\_NONE, 634  
  VTSS\_EVC\_POLICER\_ID\_DISCARD, 634  
  VTSS\_EVC\_POLICER\_ID\_EVC, 634  
  VTSS\_EVC\_POLICER\_ID\_NONE, 634  
  VTSS\_EVC\_POLICERS, 633  
vtss\_ece\_add, 639  
vtss\_ece\_counters\_clear, 641  
vtss\_ece\_counters\_get, 641  
vtss\_ece\_del, 639  
vtss\_ece\_init, 638  
vtss\_ece\_port\_t, 635  
vtss\_ece\_type\_t, 635  
vtss\_evc\_add, 637  
vtss\_evc\_counters\_clear, 640  
vtss\_evc\_counters\_get, 640  
vtss\_evc\_del, 638  
vtss\_evc\_get, 638  
vtss\_evc\_policer\_conf\_get, 636  
vtss\_evc\_policer\_conf\_set, 637  
vtss\_evc\_port\_conf\_get, 635  
vtss\_evc\_port\_conf\_set, 636  
vtss\_evc\_conf\_t, 94  
  learning, 95  
  network, 95  
  pb, 95  
  policer\_id, 94  
vtss\_evc\_counters\_clear  
  vtss\_evc\_api.h, 640  
vtss\_evc\_counters\_get  
  vtss\_evc\_api.h, 640  
vtss\_evc\_counters\_t, 95  
  rx\_discard, 96  
  rx\_green, 96  
  rx\_red, 96  
  rx\_yellow, 96  
  tx\_discard, 97  
  tx\_green, 97  
  tx\_yellow, 97  
vtss\_evc\_del  
  vtss\_evc\_api.h, 638  
vtss\_evc\_get  
  vtss\_evc\_api.h, 638  
vtss\_evc\_pb\_conf\_t, 97  
  ivid, 98  
  leaf, 99  
  leaf\_ivid, 98  
  leaf\_vid, 99  
  nni, 98  
  vid, 98  
vtss\_evc\_policer\_conf\_get  
  vtss\_evc\_api.h, 636  
vtss\_evc\_policer\_conf\_set  
  vtss\_evc\_api.h, 637  
vtss\_evc\_port\_conf\_get  
  vtss\_evc\_api.h, 635  
vtss\_evc\_port\_conf\_set  
  vtss\_evc\_api.h, 636  
vtss\_evc\_port\_conf\_t, 99  
  del\_coloring, 100  
vtss\_ewis\_aisl\_cons\_act\_s, 100  
  ais\_on\_lof, 100  
  ais\_on\_los, 100  
vtss\_ewis\_conf\_s, 101  
  ewis\_cntr\_thresh\_conf, 104  
  ewis\_init\_done, 101  
  ewis\_mode, 102  
  exp\_sl, 103  
  force\_mode, 102  
  path\_txti, 103  
  perf\_mode, 104  
  section\_cons\_act, 102  
  section\_txti, 102  
  static\_conf, 102  
  test\_conf, 103  
  tx\_oh, 103  
  tx\_oh\_passthru, 103  
vtss\_ewis\_cons\_act\_get

vtss\_wis\_api.h, 1152  
 vtss\_ewis\_cons\_act\_s, 104  
     aisl, 105  
     fault, 105  
     rdil, 105  
 vtss\_ewis\_cons\_act\_set  
     vtss\_wis\_api.h, 1152  
 vtss\_ewis\_counter\_get  
     vtss\_wis\_api.h, 1160  
 vtss\_ewis\_counter\_s, 105  
     pf\_ebc\_l, 106  
     pf\_ebc\_p, 106  
     pn\_ebc\_l, 106  
     pn\_ebc\_p, 106  
     pn\_ebc\_s, 106  
 vtss\_ewis\_counter\_threshold\_get  
     vtss\_wis\_api.h, 1161  
 vtss\_ewis\_counter\_threshold\_s, 107  
     f\_ebc\_thr\_l, 108  
     f\_ebc\_thr\_p, 108  
     n\_ebc\_thr\_l, 107  
     n\_ebc\_thr\_p, 108  
     n\_ebc\_thr\_s, 107  
 vtss\_ewis\_counter\_threshold\_set  
     vtss\_wis\_api.h, 1160  
 vtss\_ewis\_defects\_get  
     vtss\_wis\_api.h, 1158  
 vtss\_ewis\_defects\_s, 108  
     dais\_l, 110  
     dais\_p, 110  
     dfais\_p, 111  
     dfplm\_p, 111  
     dfuneq\_p, 112  
     dlcd\_p, 111  
     dlof\_s, 109  
     dlop\_p, 110  
     dlos\_s, 109  
     doof\_s, 109  
     dplm\_p, 111  
     drdi\_l, 110  
     drdi\_p, 111  
     duneq\_p, 110  
 vtss\_ewis\_event\_enable  
     vtss\_wis\_api.h, 1145  
 vtss\_ewis\_event\_force  
     vtss\_wis\_api.h, 1147  
 vtss\_ewis\_event\_poll  
     vtss\_wis\_api.h, 1146  
 vtss\_ewis\_event\_poll\_without\_mask  
     vtss\_wis\_api.h, 1146  
 vtss\_ewis\_event\_t  
     vtss\_wis\_api.h, 1143  
 vtss\_ewis\_exp\_sl\_set  
     vtss\_wis\_api.h, 1153  
 vtss\_ewis\_fault\_cons\_act\_s, 112  
     fault\_on\_aisl, 114  
     fault\_on\_aisp, 114  
     fault\_on\_feaisp, 113  
         fault\_on\_feplmp, 113  
         fault\_on\_lcdp, 114  
         fault\_on\_lof, 113  
         fault\_on\_lopp, 114  
         fault\_on\_los, 113  
         fault\_on\_plmp, 114  
         fault\_on\_rdil, 113  
         fault\_on\_sef, 113  
 vtss\_ewis\_force\_conf\_get  
     vtss\_wis\_api.h, 1148  
 vtss\_ewis\_force\_conf\_set  
     vtss\_wis\_api.h, 1148  
 vtss\_ewis\_force\_mode\_s, 115  
     line\_rx\_force, 115  
     line\_tx\_force, 115  
     path\_force, 116  
 vtss\_ewis\_line\_force\_mode\_s, 116  
     force\_ais, 116  
     force\_rdi, 117  
 vtss\_ewis\_line\_tx\_force\_mode\_s, 117  
     force\_ais, 117  
     force\_rdi, 118  
 vtss\_ewis\_mode\_get  
     vtss\_wis\_api.h, 1151  
 vtss\_ewis\_mode\_set  
     vtss\_wis\_api.h, 1150  
 vtss\_ewis\_mode\_t  
     vtss\_wis\_api.h, 1144  
 vtss\_ewis\_path\_acti\_get  
     vtss\_wis\_api.h, 1159  
 vtss\_ewis\_path\_force\_mode\_s, 118  
     force\_rdi, 119  
     force\_uneq, 118  
 vtss\_ewis\_path\_txti\_get  
     vtss\_wis\_api.h, 1154  
 vtss\_ewis\_path\_txti\_set  
     vtss\_wis\_api.h, 1154  
 vtss\_ewis\_perf\_cntr\_mode\_t  
     vtss\_wis\_api.h, 1144  
 vtss\_ewis\_perf\_get  
     vtss\_wis\_api.h, 1160  
 vtss\_ewis\_perf\_mode\_get  
     vtss\_wis\_api.h, 1163  
 vtss\_ewis\_perf\_mode\_s, 119  
     pf\_ebc\_mode\_l, 120  
     pf\_ebc\_mode\_p, 120  
     pn\_ebc\_mode\_l, 120  
     pn\_ebc\_mode\_p, 120  
     pn\_ebc\_mode\_s, 119  
 vtss\_ewis\_perf\_mode\_set  
     vtss\_wis\_api.h, 1161  
 vtss\_ewis\_perf\_s, 121  
     pf\_ebc\_l, 121  
     pf\_ebc\_p, 122  
     pn\_ebc\_l, 121  
     pn\_ebc\_p, 122  
     pn\_ebc\_s, 121  
 vtss\_ewis\_prbs31\_err\_inj\_set

vtss\_wis\_api.h, 1157  
vtss\_ewis\_prbs31\_err\_inj\_t  
    vtss\_wis\_api.h, 1145  
vtss\_ewis\_rdil\_cons\_act\_s, 122  
    rdil\_on\_ais\_l, 123  
    rdil\_on\_lof, 123  
    rdil\_on\_lopc, 123  
    rdil\_on\_los, 123  
vtss\_ewis\_reset  
    vtss\_wis\_api.h, 1151  
vtss\_ewis\_section\_acti\_get  
    vtss\_wis\_api.h, 1159  
vtss\_ewis\_section\_txti\_get  
    vtss\_wis\_api.h, 1153  
vtss\_ewis\_section\_txti\_set  
    vtss\_wis\_api.h, 1153  
vtss\_ewis\_sl\_conf\_s, 124  
    exsl, 124  
vtss\_ewis\_static\_conf\_get  
    vtss\_wis\_api.h, 1147  
vtss\_ewis\_static\_conf\_s, 124  
    ewis\_cnt\_cfg, 128  
    ewis\_lof\_ctrl1, 127  
    ewis\_lof\_ctrl2, 127  
    ewis\_mode\_ctrl, 126  
    ewis\_pmtick\_ctrl, 128  
    ewis\_rx\_ctrl1, 125  
    ewis\_rx\_err\_frc1, 127  
    ewis\_rx\_frm\_ctrl1, 127  
    ewis\_rx\_frm\_ctrl2, 127  
    ewis\_tx\_a1\_a2, 126  
    ewis\_tx\_c2\_h1, 126  
    ewis\_tx\_h2\_h3, 126  
    ewis\_tx\_z0\_e1, 126  
    ewis\_txctrl1, 125  
    ewis\_txctrl2, 125  
vtss\_ewis\_static\_conf\_t  
    vtss\_wis\_api.h, 1143  
vtss\_ewis\_status\_get  
    vtss\_wis\_api.h, 1158  
vtss\_ewis\_status\_s, 128  
    fault, 129  
    link\_stat, 129  
vtss\_ewis\_test\_conf\_s, 129  
    loopback, 130  
    test\_pattern\_ana, 130  
    test\_pattern\_gen, 130  
vtss\_ewis\_test\_counter\_get  
    vtss\_wis\_api.h, 1157  
vtss\_ewis\_test\_mode\_get  
    vtss\_wis\_api.h, 1156  
vtss\_ewis\_test\_mode\_set  
    vtss\_wis\_api.h, 1156  
vtss\_ewis\_test\_pattern\_s  
    vtss\_wis\_api.h, 1144  
vtss\_ewis\_test\_status\_s, 130  
    ana\_sync, 131  
    tstpat\_cnt, 131  
vtss\_ewis\_tti\_mode\_t  
    vtss\_wis\_api.h, 1143  
vtss\_ewis\_tti\_s, 131  
    mode, 132  
    tti, 132  
    valid, 132  
vtss\_ewis\_tx\_oh\_get  
    vtss\_wis\_api.h, 1149  
vtss\_ewis\_tx\_oh\_passthru\_get  
    vtss\_wis\_api.h, 1150  
vtss\_ewis\_tx\_oh\_passthru\_set  
    vtss\_wis\_api.h, 1149  
vtss\_ewis\_tx\_oh\_s, 132  
    tx\_c2, 135  
    tx\_dcc\_l, 134  
    tx\_dcc\_s, 133  
    tx\_e1, 133  
    tx\_e2, 134  
    tx\_f1, 133  
    tx\_f2, 135  
    tx\_k1\_k2, 134  
    tx\_n1, 135  
    tx\_s1, 134  
    tx\_z0, 134  
    tx\_z1\_z2, 135  
    tx\_z3\_z4, 135  
vtss\_ewis\_tx\_oh\_set  
    vtss\_wis\_api.h, 1149  
vtss\_ewis\_tx\_passthru\_s, 136  
    tx\_b1, 137  
    tx\_b2, 138  
    tx\_dcc\_l, 139  
    tx\_dcc\_s, 137  
    tx\_e1, 137  
    tx\_e2, 139  
    tx\_f1, 137  
    tx\_j0, 136  
    tx\_k1, 138  
    tx\_k2, 138  
    tx\_loh, 139  
    tx\_reil, 138  
    tx\_s1, 139  
    tx\_soh, 138  
    tx\_z0, 137  
    tx\_z1\_z2, 139  
vtss\_fan\_conf\_t, 140  
    fan\_low\_pol, 140  
    fan\_open\_col, 141  
    fan\_pwm\_freq, 140  
    ppr, 141  
    type, 141  
vtss\_fan\_controller\_init  
    vtss\_misc\_api.h, 782  
vtss\_fan\_cool\_lv\_get  
    vtss\_misc\_api.h, 783  
vtss\_fan\_cool\_lv\_set  
    vtss\_misc\_api.h, 782  
vtss\_fan\_rotation\_get

vtss\_misc\_api.h, 781  
 vtss\_fdma\_afi\_cancel  
     vtss\_fdma\_api.h, 654  
 vtss\_fdma\_afi\_frm\_cnt  
     vtss\_fdma\_api.h, 654  
 vtss\_fdma\_afi\_type\_t  
     vtss\_fdma\_api.h, 649  
 vtss\_fdma\_api.h  
     VTSS\_AFI\_FPS\_MAX, 646  
     VTSS\_FDMA\_CCM\_FPS\_MAX, 646  
     VTSS\_FDMA\_CCM\_FREQ\_LIST\_LEN, 646  
     VTSS\_FDMA\_CCM\_QUOTIENT\_MAX, 646  
     VTSS\_FDMA\_CH\_CNT, 643  
     VTSS\_FDMA\_DCB\_SIZE\_BYTES, 644  
     VTSS\_FDMA\_HDR\_SIZE\_BYTES, 644  
     VTSS\_FDMA\_MAX\_DATA\_PER\_DCB\_BYTES,  
         644  
     VTSS\_FDMA\_MAX\_FRAME\_SIZE\_BYTES, 645  
     VTSS\_FDMA\_MIN\_DATA\_PER\_INJ\_SOF\_DC←  
         B\_BYTES, 645  
     VTSS\_FDMA\_MIN\_DATA\_PER\_NON\_SOF\_D←  
         CB\_BYTES, 645  
     VTSS\_FDMA\_MIN\_DATA\_PER\_XTR\_SOF\_DC←  
         B\_BYTES, 645  
     VTSS\_FDMA\_MIN\_FRAME\_SIZE\_BYTES, 644  
     VTSS\_OS\_COMPILER\_ATTRIBUTE\_ALIGNED,  
         646  
     VTSS\_OS\_DCACHE\_LINE\_SIZE\_BYTES, 645  
     VTSS\_PHYS\_PORT\_CNT, 644  
     vtss\_fdma\_afi\_cancel, 654  
     vtss\_fdma\_afi\_frm\_cnt, 654  
     vtss\_fdma\_afi\_type\_t, 649  
     vtss\_fdma\_cfg, 651  
     vtss\_fdma\_ch\_t, 647  
     vtss\_fdma\_ch\_usage\_t, 650  
     vtss\_fdma\_dcb\_get, 655  
     vtss\_fdma\_dcb\_release, 652  
     vtss\_fdma\_dcb\_type\_t, 650  
     vtss\_fdma\_irq\_handler, 658  
     vtss\_fdma\_list\_t, 649  
     vtss\_fdma\_stats\_clr, 657  
     vtss\_fdma\_throttle\_cfg\_get, 655  
     vtss\_fdma\_throttle\_cfg\_set, 656  
     vtss\_fdma\_throttle\_tick, 657  
     vtss\_fdma\_tx, 652  
     vtss\_fdma\_tx\_info\_init, 653  
     vtss\_fdma\_uninit, 650  
 vtss\_fdma\_cfg  
     vtss\_fdma\_api.h, 651  
 vtss\_fdma\_cfg\_t, 141  
     afi\_buf\_cnt, 146  
     afi\_done\_cb, 146  
     enable, 142  
     rx\_alloc\_cb, 143  
     rx\_allow\_multiple\_dcbs, 145  
     rx\_allow\_vlan\_tag\_mismatch, 144  
     rx\_buf\_cnt, 143  
     rx\_cb, 145  
     rx\_dont\_reinsert\_vlan\_tag, 144  
     rx\_dont\_strip\_vlan\_tag, 144  
     rx\_mtu, 142  
     tx\_buf\_cnt, 145  
     tx\_done\_cb, 146  
 vtss\_fdma\_ch\_cfg\_t, 147  
     ccm\_quotient\_max, 150  
     chip\_no, 150  
     inj\_grp\_mask, 148  
     list, 148  
     prio, 149  
     usage, 148  
     xtr\_cb, 149  
     xtr\_grp, 148  
 vtss\_fdma\_ch\_t  
     vtss\_fdma\_api.h, 647  
 vtss\_fdma\_ch\_usage\_t  
     vtss\_fdma\_api.h, 650  
 vtss\_fdma\_dcb\_get  
     vtss\_fdma\_api.h, 655  
 vtss\_fdma\_dcb\_release  
     vtss\_fdma\_api.h, 652  
 vtss\_fdma\_dcb\_type\_t  
     vtss\_fdma\_api.h, 650  
 vtss\_fdma\_irq\_handler  
     vtss\_fdma\_api.h, 658  
 vtss\_fdma\_list\_t  
     vtss\_fdma\_api.h, 649  
 vtss\_fdma\_stats\_clr  
     vtss\_fdma\_api.h, 657  
 vtss\_fdma\_throttle\_cfg\_get  
     vtss\_fdma\_api.h, 655  
 vtss\_fdma\_throttle\_cfg\_set  
     vtss\_fdma\_api.h, 656  
 vtss\_fdma\_throttle\_cfg\_t, 151  
     byte\_limit\_per\_tick, 151  
     frm\_limit\_per\_tick, 151  
     suspend\_tick\_cnt, 152  
 vtss\_fdma\_throttle\_tick  
     vtss\_fdma\_api.h, 657  
 vtss\_fdma\_tx  
     vtss\_fdma\_api.h, 652  
 vtss\_fdma\_tx\_info\_init  
     vtss\_fdma\_api.h, 653  
 vtss\_fdma\_tx\_info\_t, 152  
     afi\_enable\_counting, 154  
     afi\_enable\_sequence\_numbering, 154  
     afi\_fps, 153  
     afi\_sequence\_number\_offset, 155  
     afi\_type, 154  
     pre\_cb, 153  
     pre\_cb\_ctxt1, 153  
     pre\_cb\_ctxt2, 153  
 vtss\_fdma\_uninit  
     vtss\_fdma\_api.h, 650  
 vtss\_fefi\_mode\_t  
     vtss\_phy\_api.h, 957  
 vtss\_fiber\_port\_speed\_t

port.h, 592  
vtss\_gpio\_10g\_aggr\_intrpt\_t  
  vtss\_phy\_10g\_api.h, 880  
vtss\_gpio\_10g\_chan\_intrpt\_t  
  vtss\_phy\_10g\_api.h, 879  
vtss\_gpio\_10g\_gpio\_intr\_sgnl\_t  
  vtss\_phy\_10g\_api.h, 877  
vtss\_gpio\_10g\_gpio\_mode\_t, 156  
  aggr\_intrpt, 157  
  c\_intrpt, 157  
  in\_sig, 157  
  input, 156  
  invert\_output, 158  
  led\_conf, 158  
  mode, 156  
  p\_gpio, 157  
  p\_gpio\_intrpt, 158  
  port, 156  
  source, 157  
  use\_as\_intrpt, 158  
vtss\_gpio\_10g\_input\_t  
  vtss\_phy\_10g\_api.h, 881  
vtss\_gpio\_10g\_led\_blink\_t  
  vtss\_phy\_10g\_api.h, 882  
vtss\_gpio\_10g\_led\_conf\_t, 159  
  blink, 159  
vtss\_gpio\_10g\_led\_mode\_t  
  vtss\_phy\_10g\_api.h, 881  
vtss\_gpio\_10g\_no\_t  
  vtss\_phy\_10g\_api.h, 861  
vtss\_gpio\_direction\_set  
  vtss\_misc\_api.h, 773  
vtss\_gpio\_event\_enable  
  vtss\_misc\_api.h, 775  
vtss\_gpio\_event\_poll  
  vtss\_misc\_api.h, 775  
vtss\_gpio\_mode\_set  
  vtss\_misc\_api.h, 773  
vtss\_gpio\_mode\_t  
  vtss\_misc\_api.h, 762  
vtss\_gpio\_read  
  vtss\_misc\_api.h, 774  
vtss\_gpio\_write  
  vtss\_misc\_api.h, 774  
vtss\_hqos\_sch\_mode\_t  
  types.h, 630  
vtss\_i2c\_read\_t  
  vtss\_init\_api.h, 664  
vtss\_i2c\_write\_t  
  vtss\_init\_api.h, 664  
vtss\_init\_api.h  
  VTSS\_I2C\_NO\_MUX, 662  
  VTSS\_QS\_CONF\_MAX, 662  
  VTSS\_QS\_CONF\_MIN, 663  
  vtss\_i2c\_read\_t, 664  
  vtss\_i2c\_write\_t, 664  
  vtss\_init\_conf\_get, 673  
  vtss\_init\_conf\_set, 673  
vtss\_inst\_create, 671  
vtss\_inst\_destroy, 671  
vtss\_inst\_get, 671  
vtss\_miim\_read\_t, 666  
vtss\_miim\_write\_t, 667  
vtss\_mmd\_read\_inc\_t, 668  
vtss\_mmd\_read\_t, 667  
vtss\_mmd\_write\_t, 668  
vtss\_port\_mux\_mode\_t, 670  
vtss\_qs\_conf\_get, 675  
vtss\_qs\_conf\_set, 675  
vtss\_qs\_mode\_t, 670  
vtss\_reg\_read\_t, 663  
vtss\_reg\_write\_t, 663  
vtss\_restart\_conf\_end, 673  
vtss\_restart\_conf\_get, 674  
vtss\_restart\_conf\_set, 675  
vtss\_restart\_status\_get, 674  
vtss\_restart\_t, 670  
vtss\_spi\_32bit\_read\_write\_t, 665  
vtss\_spi\_64bit\_read\_write\_t, 666  
vtss\_spi\_read\_write\_t, 665  
vtss\_target\_type\_t, 669  
vtss\_init\_conf\_get  
  vtss\_init\_api.h, 673  
vtss\_init\_conf\_set  
  vtss\_init\_api.h, 673  
vtss\_init\_conf\_t, 159  
  miim\_read, 160  
  miim\_write, 161  
  mmd\_read, 161  
  mmd\_read\_inc, 161  
  mmd\_write, 161  
  mux\_mode, 163  
  pi, 163  
  qs\_conf, 163  
  reg\_read, 160  
  reg\_write, 160  
  restart\_info\_port, 162  
  restart\_info\_src, 162  
  serdes, 163  
  spi\_32bit\_read\_write, 162  
  spi\_64bit\_read\_write, 162  
  spi\_read\_write, 161  
  warm\_start\_enable, 162  
vtss\_inst\_create  
  vtss\_init\_api.h, 671  
vtss\_inst\_create\_t, 164  
  target, 164  
vtss\_inst\_destroy  
  vtss\_init\_api.h, 671  
vtss\_inst\_get  
  vtss\_init\_api.h, 671  
vtss\_internal\_bw\_t  
  vtss\_port\_api.h, 1079  
vtss\_intr\_cfg  
  vtss\_misc\_api.h, 778  
vtss\_intr\_sticky\_clear

vtss\_misc\_api.h, 770  
 vtss\_ip\_addr\_t, 164  
   addr, 165  
   ipv4, 165  
   ipv6, 165  
   type, 165  
 vtss\_ip\_network\_t, 166  
   address, 166  
   prefix\_size, 166  
 vtss\_ip\_type\_t  
   types.h, 627  
 vtss\_ipv4\_mc\_flood\_members\_get  
   vtss\_l2\_api.h, 733  
 vtss\_ipv4\_mc\_flood\_members\_set  
   vtss\_l2\_api.h, 733  
 vtss\_ipv4\_network\_t, 166  
   address, 167  
   prefix\_size, 167  
 vtss\_ipv4\_uc\_t, 167  
   destination, 168  
   network, 168  
 vtss\_ipv6\_mc\_ctrl\_flood\_get  
   vtss\_l2\_api.h, 734  
 vtss\_ipv6\_mc\_ctrl\_flood\_set  
   vtss\_l2\_api.h, 735  
 vtss\_ipv6\_mc\_flood\_members\_get  
   vtss\_l2\_api.h, 733  
 vtss\_ipv6\_mc\_flood\_members\_set  
   vtss\_l2\_api.h, 734  
 vtss\_ipv6\_network\_t, 168  
   address, 169  
   prefix\_size, 169  
 vtss\_ipv6\_t, 169  
   addr, 170  
 vtss\_ipv6\_uc\_t, 170  
   destination, 170  
   network, 170  
 vtss\_irq\_conf\_get  
   vtss\_misc\_api.h, 778  
 vtss\_irq\_conf\_set  
   vtss\_misc\_api.h, 779  
 vtss\_irq\_conf\_t, 171  
   destination, 171  
   external, 171  
 vtss\_irq\_enable  
   vtss\_misc\_api.h, 780  
 vtss\_irq\_status\_get\_and\_mask  
   vtss\_misc\_api.h, 779  
 vtss\_irq\_status\_t, 172  
   active, 172  
   raw\_ident, 172  
   raw\_mask, 173  
   raw\_status, 173  
 vtss\_irq\_t  
   vtss\_misc\_api.h, 763  
 vtss\_isidx\_t  
   types.h, 621  
 vtss\_isolated\_port\_members\_get  
   vtss\_l2\_api.h, 718  
 vtss\_isolated\_port\_members\_set  
   vtss\_l2\_api.h, 718  
 vtss\_isolated\_vlan\_get  
   vtss\_l2\_api.h, 717  
 vtss\_isolated\_vlan\_set  
   vtss\_l2\_api.h, 717  
 vtss\_l2\_api.h  
   VTSS\_ERPI\_ARRAY\_SIZE, 692  
   VTSS\_ERPI\_END, 692  
   VTSS\_ERPI\_START, 691  
   VTSS\_ERPIS, 691  
   VTSS\_MAC\_ADDRS, 684  
   VTSS\_MSTI\_ARRAY\_SIZE, 687  
   VTSS\_MSTI\_END, 686  
   VTSS\_MSTI\_START, 686  
   VTSS\_MSTIS, 686  
   VTSS\_PVLAN\_ARRAY\_SIZE, 691  
   VTSS\_PVLAN\_NO\_DEFAULT, 691  
   VTSS\_PVLAN\_NO\_END, 691  
   VTSS\_PVLAN\_NO\_START, 690  
   VTSS\_PVLANS, 690  
   VTSS\_UPSPN\_CPU, 686  
   VTSS\_UPSPN\_NONE, 686  
   VTSS\_VCE\_ID\_LAST, 688  
   VTSS\_VCL\_ARRAY\_SIZE, 687  
   VTSS\_VCL\_ID\_END, 687  
   VTSS\_VCL\_ID\_START, 687  
   VTSS\_VCL\_IDS, 687  
   VTSS\_VLAN\_TRANS\_FIRST\_GROUP\_ID, 688  
   VTSS\_VLAN\_TRANS\_GROUP\_MAX\_CNT, 688  
   VTSS\_VLAN\_TRANS\_LAST\_GROUP\_ID, 689  
   VTSS\_VLAN\_TRANS\_MAX\_CNT, 688  
   VTSS\_VLAN\_TRANS\_MAX\_VLAN\_ID, 689  
   VTSS\_VLAN\_TRANS\_NULL\_CHECK, 690  
   VTSS\_VLAN\_TRANS\_NULL\_GROUP\_ID, 688  
   VTSS\_VLAN\_TRANS\_PORT\_BF\_SIZE, 690  
   VTSS\_VLAN\_TRANS\_VALID\_GROUP\_CHECK,  
     689  
   VTSS\_VLAN\_TRANS\_VALID\_VLAN\_CHECK,  
     689  
   VTSS\_VLAN\_TRANS\_VID\_START, 689  
   VTSS\_VSTAX\_UPSID\_LEGAL, 685  
   VTSS\_VSTAX\_UPSID\_MAX, 685  
   VTSS\_VSTAX\_UPSID\_MIN, 685  
   VTSS\_VSTAX\_UPSID\_START, 685  
   VTSS\_VSTAX\_UPSID\_UNDEF, 685  
   VTSS\_VSTAX\_UPSIDS, 684  
   vtss\_aggr\_glag\_members\_get, 724  
   vtss\_aggr\_mode\_get, 724  
   vtss\_aggr\_mode\_set, 724  
   vtss\_aggr\_port\_members\_get, 723  
   vtss\_aggr\_port\_members\_set, 723  
   vtss\_apvlan\_port\_members\_get, 720  
   vtss\_apvlan\_port\_members\_set, 720  
   vtss\_dgroup\_port\_conf\_get, 720  
   vtss\_dgroup\_port\_conf\_set, 721  
   vtss\_eps\_port\_conf\_get, 735

vtss\_eps\_port\_conf\_set, 735  
vtss\_eps\_port\_selector\_get, 736  
vtss\_eps\_port\_selector\_set, 736  
vtss\_eps\_port\_type\_t, 694  
vtss\_eps\_selector\_t, 694  
vtss\_erps\_port\_state\_get, 738  
vtss\_erps\_port\_state\_set, 738  
vtss\_erps\_state\_t, 695  
vtss\_erps\_vlan\_member\_get, 737  
vtss\_erps\_vlan\_member\_set, 737  
vtss\_ipv4\_mc\_flood\_members\_get, 733  
vtss\_ipv4\_mc\_flood\_members\_set, 733  
vtss\_ipv6\_mc\_ctrl\_flood\_get, 734  
vtss\_ipv6\_mc\_ctrl\_flood\_set, 735  
vtss\_ipv6\_mc\_flood\_members\_get, 733  
vtss\_ipv6\_mc\_flood\_members\_set, 734  
vtss\_isolated\_port\_members\_get, 718  
vtss\_isolated\_port\_members\_set, 718  
vtss\_isolated\_vlan\_get, 717  
vtss\_isolated\_vlan\_set, 717  
vtss\_learn\_port\_mode\_get, 703  
vtss\_learn\_port\_mode\_set, 704  
vtss\_mac\_table\_add, 695  
vtss\_mac\_table\_age, 699  
vtss\_mac\_table\_age\_time\_get, 698  
vtss\_mac\_table\_age\_time\_set, 698  
vtss\_mac\_table\_del, 697  
vtss\_mac\_table\_flush, 699  
vtss\_mac\_table\_get, 697  
vtss\_mac\_table\_get\_next, 698  
vtss\_mac\_table\_glag\_add, 702  
vtss\_mac\_table\_glag\_flush, 702  
vtss\_mac\_table\_port\_flush, 700  
vtss\_mac\_table\_status\_get, 703  
vtss\_mac\_table\_upsid\_flush, 701  
vtss\_mac\_table\_upsid\_upspn\_flush, 701  
vtss\_mac\_table\_vlan\_age, 699  
vtss\_mac\_table\_vlan\_flush, 700  
vtss\_mac\_table\_vlan\_glag\_flush, 703  
vtss\_mac\_table\_vlan\_port\_flush, 701  
vtss\_mc\_flood\_members\_get, 732  
vtss\_mc\_flood\_members\_set, 732  
vtss\_mirror\_conf\_get, 727  
vtss\_mirror\_conf\_set, 727  
vtss\_mirror\_cpu\_egress\_get, 730  
vtss\_mirror\_cpu\_egress\_set, 731  
vtss\_mirror\_cpu\_ingress\_get, 730  
vtss\_mirror\_cpu\_ingress\_set, 730  
vtss\_mirror\_egress\_ports\_get, 729  
vtss\_mirror\_egress\_ports\_set, 729  
vtss\_mirror\_ingress\_ports\_get, 728  
vtss\_mirror\_ingress\_ports\_set, 728  
vtss\_mirror\_monitor\_port\_get, 727  
vtss\_mirror\_monitor\_port\_set, 728  
vtss\_mirror\_tag\_t, 694  
vtss\_mstp\_port\_msti\_state\_get, 707  
vtss\_mstp\_port\_msti\_state\_set, 707  
vtss\_mstp\_vlan\_msti\_get, 706  
vtss\_mstp\_vlan\_msti\_set, 706  
vtss\_port\_state\_get, 704  
vtss\_port\_state\_set, 705  
vtss\_pvlan\_port\_members\_get, 719  
vtss\_pvlan\_port\_members\_set, 719  
vtss\_sflow\_port\_conf\_get, 721  
vtss\_sflow\_port\_conf\_set, 722  
vtss\_sflow\_sampling\_rate\_convert, 722  
vtss\_stp\_port\_state\_get, 705  
vtss\_stp\_port\_state\_set, 705  
vtss\_stp\_state\_t, 692  
vtss\_uc\_flood\_members\_get, 731  
vtss\_uc\_flood\_members\_set, 732  
vtss\_vce\_add, 714  
vtss\_vce\_del, 714  
vtss\_vce\_init, 714  
vtss\_vce\_type\_t, 693  
vtss\_vcl\_port\_conf\_get, 713  
vtss\_vcl\_port\_conf\_set, 713  
vtss\_vlan\_conf\_get, 708  
vtss\_vlan\_conf\_set, 708  
vtss\_vlan\_port\_conf\_get, 708  
vtss\_vlan\_port\_conf\_set, 710  
vtss\_vlan\_port\_members\_get, 710  
vtss\_vlan\_port\_members\_set, 711  
vtss\_vlan\_port\_type\_t, 693  
vtss\_vlan\_trans\_group\_add, 715  
vtss\_vlan\_trans\_group\_del, 715  
vtss\_vlan\_trans\_group\_get, 716  
vtss\_vlan\_trans\_group\_to\_port\_get, 717  
vtss\_vlan\_trans\_group\_to\_port\_set, 716  
vtss\_vlan\_tx\_tag\_get, 712  
vtss\_vlan\_tx\_tag\_set, 712  
vtss\_vlan\_tx\_tag\_t, 693  
vtss\_vlan\_vid\_conf\_get, 711  
vtss\_vlan\_vid\_conf\_set, 711  
vtss\_vstax\_conf\_get, 738  
vtss\_vstax\_conf\_set, 739  
vtss\_vstax\_glag\_get, 726  
vtss\_vstax\_glag\_set, 726  
vtss\_vstax\_master\_upsid\_get, 740  
vtss\_vstax\_master\_upsid\_set, 741  
vtss\_vstax\_port\_conf\_get, 739  
vtss\_vstax\_port\_conf\_set, 740  
vtss\_vstax\_topology\_set, 741  
vtss\_vstax\_topology\_type\_t, 695  
vtss\_vt\_id\_t, 692  
vtss\_l3\_api.h  
VTSS\_ARP\_CNT, 744  
VTSS\_ARP\_IPV4\_RELATIONS, 745  
VTSS\_ARP\_IPV6\_RELATIONS, 745  
VTSS\_JR1\_ARP\_CNT, 744  
VTSS\_JR1\_LPM\_CNT, 743  
VTSS\_JR1\_RLEG\_CNT, 744  
VTSS\_LPM\_CNT, 744  
VTSS\_RLEG\_CNT, 744  
vtss\_l3\_common\_get, 746  
vtss\_l3\_common\_set, 747

vtss\_I3\_counters\_reset, 751  
 vtss\_I3\_counters\_rleg\_clear, 752  
 vtss\_I3\_counters\_rleg\_get, 752  
 vtss\_I3\_counters\_system\_get, 752  
 vtss\_I3\_flush, 746  
 vtss\_I3\_neighbour\_add, 751  
 vtss\_I3\_neighbour\_del, 751  
 vtss\_I3\_neighbour\_get, 750  
 vtss\_I3\_neighbour\_type\_t, 745  
 vtss\_I3\_rleg\_add, 748  
 vtss\_I3\_rleg\_common\_mode\_t, 745  
 vtss\_I3\_rleg\_del, 749  
 vtss\_I3\_rleg\_get, 747  
 vtss\_I3\_rleg\_get\_specific, 747  
 vtss\_I3\_rleg\_update, 748  
 vtss\_I3\_route\_add, 749  
 vtss\_I3\_route\_del, 750  
 vtss\_I3\_route\_get, 749  
 vtss\_I3\_common\_conf\_t, 173  
     base\_address, 174  
     rleg\_mode, 174  
     routing\_enable, 174  
 vtss\_I3\_common\_get  
     vtss\_I3\_api.h, 746  
 vtss\_I3\_common\_set  
     vtss\_I3\_api.h, 747  
 vtss\_I3\_counters\_reset  
     vtss\_I3\_api.h, 751  
 vtss\_I3\_counters\_rleg\_clear  
     vtss\_I3\_api.h, 752  
 vtss\_I3\_counters\_rleg\_get  
     vtss\_I3\_api.h, 752  
 vtss\_I3\_counters\_system\_get  
     vtss\_I3\_api.h, 752  
 vtss\_I3\_counters\_t, 174  
     ipv4uc\_received\_frames, 175  
     ipv4uc\_received\_octets, 175  
     ipv4uc\_transmitted\_frames, 176  
     ipv4uc\_transmitted\_octets, 175  
     ipv6uc\_received\_frames, 175  
     ipv6uc\_received\_octets, 175  
     ipv6uc\_transmitted\_frames, 176  
     ipv6uc\_transmitted\_octets, 176  
 vtss\_I3\_flush  
     vtss\_I3\_api.h, 746  
 vtss\_I3\_neighbour\_add  
     vtss\_I3\_api.h, 751  
 vtss\_I3\_neighbour\_del  
     vtss\_I3\_api.h, 751  
 vtss\_I3\_neighbour\_get  
     vtss\_I3\_api.h, 750  
 vtss\_I3\_neighbour\_t, 176  
     dip, 177  
     dmac, 177  
     vlan, 177  
 vtss\_I3\_neighbour\_type\_t  
     vtss\_I3\_api.h, 745  
 vtss\_I3\_rleg\_add  
     vtss\_I3\_api.h, 748  
 vtss\_I3\_rleg\_common\_mode\_t  
     vtss\_I3\_api.h, 745  
 vtss\_I3\_rleg\_conf\_t, 178  
     ipv4\_icmp\_redirect\_enable, 178  
     ipv4\_unicast\_enable, 178  
     ipv6\_icmp\_redirect\_enable, 179  
     ipv6\_unicast\_enable, 178  
     vlan, 179  
     vrid0, 179  
     vrid0\_enable, 179  
     vrid1, 180  
     vrid1\_enable, 180  
 vtss\_I3\_rleg\_del  
     vtss\_I3\_api.h, 749  
 vtss\_I3\_rleg\_get  
     vtss\_I3\_api.h, 747  
 vtss\_I3\_rleg\_get\_specific  
     vtss\_I3\_api.h, 747  
 vtss\_I3\_rleg\_update  
     vtss\_I3\_api.h, 748  
 vtss\_I3\_route\_add  
     vtss\_I3\_api.h, 749  
 vtss\_I3\_route\_del  
     vtss\_I3\_api.h, 750  
 vtss\_I3\_route\_get  
     vtss\_I3\_api.h, 749  
 vtss\_lb\_type\_t  
     vtss\_phy\_10g\_api.h, 873  
 vtss\_lcpll\_status\_t, 180  
     cal\_done, 181  
     cal\_error, 181  
     fsm\_lock, 181  
     fsm\_stat, 181  
     gain\_stat, 182  
     lock\_status, 181  
 vtss\_learn\_mode\_t, 182  
     automatic, 182  
     cpu, 183  
     discard, 183  
 vtss\_learn\_port\_mode\_get  
     vtss\_I2\_api.h, 703  
 vtss\_learn\_port\_mode\_set  
     vtss\_I2\_api.h, 704  
 vtss\_mac\_addr\_t  
     types.h, 621  
 vtss\_mac\_t, 183  
     addr, 184  
 vtss\_mac\_table\_add  
     vtss\_I2\_api.h, 695  
 vtss\_mac\_table\_age  
     vtss\_I2\_api.h, 699  
 vtss\_mac\_table\_age\_time\_get  
     vtss\_I2\_api.h, 698  
 vtss\_mac\_table\_age\_time\_set  
     vtss\_I2\_api.h, 698  
 vtss\_mac\_table\_del  
     vtss\_I2\_api.h, 697

vtss\_mac\_table\_entry\_t, 184  
aged, 185  
copy\_to\_cpu, 185  
cpu\_queue, 185  
destination, 185  
enable, 186  
locked, 185  
remote\_entry, 186  
upsid, 186  
upspn, 186  
vid\_mac, 184  
vstax2, 186  
vtss\_mac\_table\_flush  
    vtss\_l2\_api.h, 699  
vtss\_mac\_table\_get  
    vtss\_l2\_api.h, 697  
vtss\_mac\_table\_get\_next  
    vtss\_l2\_api.h, 698  
vtss\_mac\_table\_glag\_add  
    vtss\_l2\_api.h, 702  
vtss\_mac\_table\_glag\_flush  
    vtss\_l2\_api.h, 702  
vtss\_mac\_table\_port\_flush  
    vtss\_l2\_api.h, 700  
vtss\_mac\_table\_status\_get  
    vtss\_l2\_api.h, 703  
vtss\_mac\_table\_status\_t, 187  
    aged, 188  
    learned, 187  
    moved, 188  
    replaced, 187  
vtss\_mac\_table\_upsid\_flush  
    vtss\_l2\_api.h, 701  
vtss\_mac\_table\_upsid\_upspn\_flush  
    vtss\_l2\_api.h, 701  
vtss\_mac\_table\_vlan\_age  
    vtss\_l2\_api.h, 699  
vtss\_mac\_table\_vlan\_flush  
    vtss\_l2\_api.h, 700  
vtss\_mac\_table\_vlan\_glag\_flush  
    vtss\_l2\_api.h, 703  
vtss\_mac\_table\_vlan\_port\_flush  
    vtss\_l2\_api.h, 701  
vtss\_mc\_flood\_members\_get  
    vtss\_l2\_api.h, 732  
vtss\_mc\_flood\_members\_set  
    vtss\_l2\_api.h, 732  
vtss\_mem\_flags\_t  
    types.h, 624  
vtss\_miim\_controller\_t  
    vtss\_port\_api.h, 1078  
vtss\_miim\_read  
    vtss\_port\_api.h, 1087  
vtss\_miim\_read\_t  
    vtss\_init\_api.h, 666  
vtss\_miim\_write  
    vtss\_port\_api.h, 1087  
vtss\_miim\_write\_t  
    vtss\_init\_api.h, 667  
vtss\_mirror\_conf\_get  
    vtss\_l2\_api.h, 727  
vtss\_mirror\_conf\_set  
    vtss\_l2\_api.h, 727  
vtss\_mirror\_conf\_t, 188  
    dei, 190  
    fwd\_enable, 189  
    pcp, 189  
    port\_no, 189  
    tag, 189  
    vid, 189  
vtss\_mirror\_cpu\_egress\_get  
    vtss\_l2\_api.h, 730  
vtss\_mirror\_cpu\_egress\_set  
    vtss\_l2\_api.h, 731  
vtss\_mirror\_cpu\_ingress\_get  
    vtss\_l2\_api.h, 730  
vtss\_mirror\_cpu\_ingress\_set  
    vtss\_l2\_api.h, 730  
vtss\_mirror\_egress\_ports\_get  
    vtss\_l2\_api.h, 729  
vtss\_mirror\_egress\_ports\_set  
    vtss\_l2\_api.h, 729  
vtss\_mirror\_ingress\_ports\_get  
    vtss\_l2\_api.h, 728  
vtss\_mirror\_ingress\_ports\_set  
    vtss\_l2\_api.h, 728  
vtss\_mirror\_monitor\_port\_get  
    vtss\_l2\_api.h, 727  
vtss\_mirror\_monitor\_port\_set  
    vtss\_l2\_api.h, 728  
vtss\_mirror\_tag\_t  
    vtss\_l2\_api.h, 694  
vtss\_misc\_api.h  
    tod\_get\_ns\_cnt\_cb\_t, 759  
    VTSS\_OS\_TIMESTAMP\_TYPE, 759  
    VTSS\_OS\_TIMESTAMP, 759  
    vtss\_callout\_lock, 767  
    vtss\_callout\_trace\_hex\_dump, 766  
    vtss\_callout\_trace\_printf, 765  
    vtss\_callout\_unlock, 767  
    vtss\_chip\_id\_get, 770  
    vtss\_debug\_group\_t, 761  
    vtss\_debug\_info\_get, 766  
    vtss\_debug\_info\_print, 767  
    vtss\_debug\_layer\_t, 761  
    vtss\_debug\_lock, 768  
    vtss\_debug\_reg\_check\_set, 784  
    vtss\_debug\_unlock, 768  
    vtss\_dev\_all\_event\_enable, 772  
    vtss\_dev\_all\_event\_poll, 772  
    vtss\_eee\_port\_conf\_set, 783  
    vtss\_eee\_port\_counter\_get, 784  
    vtss\_eee\_port\_state\_set, 783  
    vtss\_eee\_state\_select\_t, 764  
    vtss\_fan\_controller\_init, 782  
    vtss\_fan\_cool\_lvl\_get, 783

vtss\_fan\_cool\_lvl\_set, 782  
 vtss\_fan\_rotation\_get, 781  
 vtss\_gpio\_direction\_set, 773  
 vtss\_gpio\_event\_enable, 775  
 vtss\_gpio\_event\_poll, 775  
 vtss\_gpio\_mode\_set, 773  
 vtss\_gpio\_mode\_t, 762  
 vtss\_gpio\_read, 774  
 vtss\_gpio\_write, 774  
 vtss\_intr\_cfg, 778  
 vtss\_intr\_sticky\_clear, 770  
 vtss\_irq\_conf\_get, 778  
 vtss\_irq\_conf\_set, 779  
 vtss\_irq\_enable, 780  
 vtss\_irq\_status\_get\_and\_mask, 779  
 vtss\_irq\_t, 763  
 vtss\_poll\_1sec, 771  
 vtss\_ptp\_event\_enable, 771  
 vtss\_ptp\_event\_poll, 771  
 vtss\_reg\_read, 768  
 vtss\_reg\_write, 769  
 vtss\_reg\_write\_masked, 769  
 vtss\_sgpio\_bmode\_t, 763  
 vtss\_sgpio\_conf\_get, 776  
 vtss\_sgpio\_conf\_set, 776  
 vtss\_sgpio\_event\_enable, 777  
 vtss\_sgpio\_event\_poll, 777  
 vtss\_sgpio\_mode\_t, 763  
 vtss\_sgpio\_read, 776  
 vtss\_temp\_sensor\_get, 781  
 vtss\_temp\_sensor\_init, 781  
 vtss\_tod\_get\_ns\_cnt, 780  
 vtss\_tod\_set\_ns\_cnt\_cb, 780  
 vtss\_trace\_conf\_get, 764  
 vtss\_trace\_conf\_set, 765  
 vtss\_trace\_group\_t, 760  
 vtss\_trace\_layer\_t, 759  
 vtss\_trace\_level\_t, 760  
 vtss\_mmd\_read  
     vtss\_port\_api.h, 1090  
 vtss\_mmd\_read\_inc\_t  
     vtss\_init\_api.h, 668  
 vtss\_mmd\_read\_t  
     vtss\_init\_api.h, 667  
 vtss\_mmd\_write  
     vtss\_port\_api.h, 1091  
 vtss\_mmd\_write\_t  
     vtss\_init\_api.h, 668  
 vtss\_mstp\_port\_msti\_state\_get  
     vtss\_l2\_api.h, 707  
 vtss\_mstp\_port\_msti\_state\_set  
     vtss\_l2\_api.h, 707  
 vtss\_mstp\_vlan\_msti\_get  
     vtss\_l2\_api.h, 706  
 vtss\_mstp\_vlan\_msti\_set  
     vtss\_l2\_api.h, 706  
 vtss\_mtimmer\_t, 190  
     now, 191  
     timeout, 190  
     vtss\_os\_custom.h, 791  
     vtss\_os\_ecos.h, 799  
 vtss\_npi\_conf\_get  
     vtss\_packet\_api.h, 818  
 vtss\_npi\_conf\_set  
     vtss\_packet\_api.h, 818  
 vtss\_npi\_conf\_t, 191  
     enable, 191  
     port\_no, 192  
 vtss\_os\_custom.h  
     uint, 787  
     ulong, 787  
     VTSS\_DIV64, 788  
     VTSS\_LABS, 789  
     VTSS\_LLabs, 789  
     VTSS\_MOD64, 788  
     VTSS\_MSLEEP, 787  
     VTSS\_MTIMER\_CANCEL, 788  
     VTSS\_MTIMER\_START, 788  
     VTSS\_MTIMER\_TIMEOUT, 788  
     VTSS\_OS\_Ctz64, 789  
     VTSS\_OS\_Ctz, 789  
     VTSS\_OS\_FREE, 790  
     VTSS\_OS\_MALLOC, 790  
     VTSS\_OS\_RAND, 790  
     vtss\_mtimmer\_t, 791  
 vtss\_os\_ecos.h  
     llabs, 800  
     VTSS\_DIV64, 793  
     VTSS\_LABS, 794  
     VTSS\_LLabs, 794  
     VTSS\_MOD64, 794  
     VTSS\_MSLEEP, 792  
     VTSS\_MTIMER\_CANCEL, 793  
     VTSS\_MTIMER\_START, 793  
     VTSS\_MTIMER\_TIMEOUT, 793  
     VTSS\_NSLEEP, 792  
     VTSS\_OS\_BIG\_ENDIAN, 797  
     VTSS\_OS\_COMPILER\_ATTRIBUTE\_ALIGNED,  
         796  
     VTSS\_OS\_Ctz64, 795  
     VTSS\_OS\_Ctz, 794  
     VTSS\_OS\_DCACHE\_FLUSH, 797  
     VTSS\_OS\_DCACHE\_INVALIDATE, 797  
     VTSS\_OS\_DCACHE\_LINE\_SIZE\_BYTES, 796  
     VTSS\_OS\_FREE, 795  
     VTSS\_OS\_INTERRUPT\_DISABLE, 799  
     VTSS\_OS\_INTERRUPT\_FLAGS, 799  
     VTSS\_OS\_INTERRUPT\_RESTORE, 799  
     VTSS\_OS\_MALLOC, 795  
     VTSS\_OS\_NTOHL, 798  
     VTSS\_OS\_RAND, 796  
     VTSS\_OS\_reordered\_barrier, 796  
     VTSS\_OS\_SCHEDULER\_FLAGS, 798  
     VTSS\_OS\_SCHEDULER\_LOCK, 798  
     VTSS\_OS\_SCHEDULER\_UNLOCK, 798  
     VTSS\_OS\_Virt\_to\_Phys, 797

VTSS\_TIME\_OF\_DAY, 793  
vtss\_callout\_free, 800  
vtss\_callout\_malloc, 800  
vtss\_mtimer\_t, 799  
vtss\_os\_linux.h  
    VTSS\_DIV64, 805  
    VTSS\_LABS, 805  
    VTSS\_LLabs, 806  
    VTSS\_MOD64, 805  
    VTSS\_MSLEEP, 803  
    VTSS\_MTIMER\_CANCEL, 804  
    VTSS\_MTIMER\_START, 803  
    VTSS\_MTIMER\_TIMEOUT, 803  
    VTSS\_NSLEEP, 802  
    VTSS\_OS\_BIG\_ENDIAN, 802  
    VTSS\_OS\_CTZ64, 806  
    VTSS\_OS\_CTZ, 806  
    VTSS\_OS\_FREE, 807  
    VTSS\_OS\_MALLOC, 807  
    VTSS\_OS\_NTOHL, 802  
    VTSS\_OS\_RAND, 807  
    VTSS\_OS\_SCHEDULER\_FLAGS, 804  
    VTSS\_OS\_SCHEDULER\_LOCK, 804  
    VTSS\_OS\_SCHEDULER\_UNLOCK, 805  
    VTSS\_TIME\_OF\_DAY, 804  
vtss\_os\_timestamp\_t, 192  
    hw\_cnt, 192  
vtss\_packet\_api.h  
    VTSS\_JR1\_PACKET\_HDR\_SIZE\_BYTES, 812  
    VTSS\_JR1\_RX\_IFH\_SIZE, 813  
    VTSS\_JR2\_PACKET\_HDR\_SIZE\_BYTES, 812  
    VTSS\_JR2\_RX\_IFH\_SIZE, 814  
    VTSS\_L26\_PACKET\_HDR\_SIZE\_BYTES, 813  
    VTSS\_L26\_RX\_IFH\_SIZE, 814  
    VTSS\_PACKET\_HDR\_SIZE\_BYTES, 813  
    VTSS\_PACKET\_TX\_IFH\_MAX, 814  
    VTSS\_PRIO\_SUPER, 812  
    VTSS\_SVL\_PACKET\_HDR\_SIZE\_BYTES, 813  
    VTSS\_SVL\_RX\_IFH\_SIZE, 813  
    VTSS\_VSTAX\_HDR\_SIZE, 812  
    VTSS\_VSTAX\_TTL\_PORT, 812  
    vtss\_npi\_conf\_get, 818  
    vtss\_npi\_conf\_set, 818  
    vtss\_packet\_dma\_conf\_get, 830  
    vtss\_packet\_dma\_conf\_set, 830  
    vtss\_packet\_dma\_offset, 830  
    vtss\_packet\_filter\_t, 814  
    vtss\_packet\_frame\_filter, 824  
    vtss\_packet\_frame\_info\_init, 824  
    vtss\_packet\_oam\_type\_t, 815  
    vtss\_packet\_port\_filter\_get, 825  
    vtss\_packet\_port\_info\_init, 825  
    vtss\_packet\_ptp\_action\_t, 816  
    vtss\_packet\_rx\_conf\_get, 819  
    vtss\_packet\_rx\_conf\_set, 819  
    vtss\_packet\_rx\_frame\_discard, 822  
    vtss\_packet\_rx\_frame\_get, 821  
    vtss\_packet\_rx\_frame\_get\_raw, 822  
        vtss\_packet\_rx\_hdr\_decode, 827  
        vtss\_packet\_rx\_hints\_t, 816  
        vtss\_packet\_rx\_port\_conf\_get, 819  
        vtss\_packet\_rx\_port\_conf\_set, 821  
        vtss\_packet\_tx\_frame, 829  
        vtss\_packet\_tx\_frame\_port, 823  
        vtss\_packet\_tx\_frame\_port\_vlan, 823  
        vtss\_packet\_tx\_frame\_vlan, 824  
        vtss\_packet\_tx\_frame\_vstax, 826  
        vtss\_packet\_tx\_hdr\_compile, 828  
        vtss\_packet\_tx\_hdr\_encode, 827  
        vtss\_packet\_tx\_info\_init, 829  
        vtss\_packet\_tx\_vstax\_t, 817  
        vtss\_packet\_vstax\_frame2header, 826  
        vtss\_packet\_vstax\_header2frame, 826  
        vtss\_tag\_type\_t, 816  
        vtss\_vstax\_fwd\_mode\_t, 815  
    vtss\_packet\_dma\_conf\_get  
        vtss\_packet\_api.h, 830  
    vtss\_packet\_dma\_conf\_set  
        vtss\_packet\_api.h, 830  
    vtss\_packet\_dma\_conf\_t, 193  
        dma\_enable, 193  
    vtss\_packet\_dma\_offset  
        vtss\_packet\_api.h, 830  
    vtss\_packet\_filter\_t  
        vtss\_packet\_api.h, 814  
    vtss\_packet\_frame\_filter  
        vtss\_packet\_api.h, 824  
    vtss\_packet\_frame\_info\_init  
        vtss\_packet\_api.h, 824  
    vtss\_packet\_frame\_info\_t, 194  
        aggr\_rx\_disable, 195  
        aggr\_tx\_disable, 195  
        glag\_no, 194  
        port\_no, 194  
        port\_tx, 195  
        vid, 194  
    vtss\_packet\_oam\_type\_t  
        vtss\_packet\_api.h, 815  
    vtss\_packet\_port\_filter\_get  
        vtss\_packet\_api.h, 825  
    vtss\_packet\_port\_filter\_t, 195  
        filter, 196  
        tpid, 196  
    vtss\_packet\_port\_info\_init  
        vtss\_packet\_api.h, 825  
    vtss\_packet\_port\_info\_t, 196  
        aggr\_rx\_disable, 197  
        aggr\_tx\_disable, 198  
        glag\_no, 197  
        port\_no, 197  
        vid, 197  
    vtss\_packet\_ptp\_action\_t  
        vtss\_packet\_api.h, 816  
    vtss\_packet\_reg\_type\_t  
        types.h, 626  
    vtss\_packet\_rx\_conf\_get

vtss\_packet\_api.h, 819  
 vtss\_packet\_rx\_conf\_set  
     vtss\_packet\_api.h, 819  
 vtss\_packet\_rx\_conf\_t, 198  
     grp\_map, 199  
     map, 199  
     queue, 198  
     reg, 199  
 vtss\_packet\_rx\_frame\_discard  
     vtss\_packet\_api.h, 822  
 vtss\_packet\_rx\_frame\_get  
     vtss\_packet\_api.h, 821  
 vtss\_packet\_rx\_frame\_get\_raw  
     vtss\_packet\_api.h, 822  
 vtss\_packet\_rx\_grp\_t  
     types.h, 621  
 vtss\_packet\_rx\_hdr\_decode  
     vtss\_packet\_api.h, 827  
 vtss\_packet\_rx\_header\_t, 199  
     arrived\_tagged, 201  
     learn, 200  
     length, 200  
     port\_no, 200  
     queue\_mask, 200  
     tag, 201  
     vstax, 201  
 vtss\_packet\_rx\_hints\_t  
     vtss\_packet\_api.h, 816  
 vtss\_packet\_rx\_info\_t, 201  
     acl\_hit, 206  
     acl\_idx, 206  
     cos, 206  
     glag\_no, 203  
     hints, 202  
     hw\_tstamp, 208  
     hw\_tstamp\_decoded, 208  
     isdx, 210  
     length, 203  
     oam\_info, 209  
     oam\_info\_decoded, 210  
     port\_no, 203  
     sflow\_port\_no, 209  
     sflow\_type, 208  
     stripped\_tag, 205  
     sw\_tstamp, 207  
     tag, 204  
     tag\_type, 204  
     tstamp\_id, 207  
     tstamp\_id\_decoded, 207  
     vstax, 210  
     xtr\_qu\_mask, 205  
 vtss\_packet\_rx\_meta\_t, 211  
     chip\_no, 212  
     etype, 213  
     fcs, 213  
     length, 214  
     no\_wait, 212  
     sw\_tstamp, 214  
                 xtr\_qu, 212  
 vtss\_packet\_rx\_port\_conf\_get  
     vtss\_packet\_api.h, 819  
 vtss\_packet\_rx\_port\_conf\_set  
     vtss\_packet\_api.h, 821  
 vtss\_packet\_rx\_port\_conf\_t, 215  
     bpdu\_reg, 216  
     garp\_reg, 216  
 vtss\_packet\_rx\_queue\_conf\_t, 216  
     npi, 217  
     size, 216  
 vtss\_packet\_rx\_queue\_map\_t, 217  
     bpdu\_queue, 218  
     garp\_queue, 218  
     igmp\_queue, 218  
     ipmc\_ctrl\_queue, 218  
     l3\_other\_queue, 219  
     l3\_uc\_queue, 219  
     learn\_queue, 218  
     lrn\_all\_queue, 219  
     mac\_vid\_queue, 218  
     sflow\_queue, 219  
     stack\_queue, 219  
 vtss\_packet\_rx\_queue\_npi\_conf\_t, 220  
     enable, 220  
 vtss\_packet\_rx\_reg\_t, 221  
     bpdu\_cpu\_only, 221  
     garp\_cpu\_only, 221  
     igmp\_cpu\_only, 222  
     ipmc\_ctrl\_cpu\_copy, 221  
     mld\_cpu\_only, 222  
 vtss\_packet\_tx\_frame  
     vtss\_packet\_api.h, 829  
 vtss\_packet\_tx\_frame\_port  
     vtss\_packet\_api.h, 823  
 vtss\_packet\_tx\_frame\_port\_vlan  
     vtss\_packet\_api.h, 823  
 vtss\_packet\_tx\_frame\_vlan  
     vtss\_packet\_api.h, 824  
 vtss\_packet\_tx\_frame\_vstax  
     vtss\_packet\_api.h, 826  
 vtss\_packet\_tx\_grp\_t  
     types.h, 621  
 vtss\_packet\_tx\_hdr\_compile  
     vtss\_packet\_api.h, 828  
 vtss\_packet\_tx\_hdr\_encode  
     vtss\_packet\_api.h, 827  
 vtss\_packet\_tx\_ifh\_t, 222  
     ifh, 223  
     length, 223  
 vtss\_packet\_tx\_info\_init  
     vtss\_packet\_api.h, 829  
 vtss\_packet\_tx\_info\_t, 223  
     agr\_code, 226  
     bin, 228  
     cos, 227  
     dp, 232  
     dst\_port\_mask, 224

frm\_len, 225  
isdx, 231  
isdx\_dont\_use, 232  
latch\_timestamp, 230  
masquerade\_port, 233  
oam\_type, 231  
pdu\_offset, 233  
ptp\_action, 229  
ptp\_id, 229  
ptp\_timestamp, 230  
switch\_frm, 224  
sym, 228  
tag, 225  
tx\_vstax\_hdr, 227  
vstax, 229  
vtss\_packet\_tx\_vstax\_t  
    vtss\_packet\_api.h, 817  
vtss\_packet\_vstax\_frame2header  
    vtss\_packet\_api.h, 826  
vtss\_packet\_vstax\_header2frame  
    vtss\_packet\_api.h, 826  
vtss\_phy\_10G\_is\_valid  
    vtss\_phy\_10g\_api.h, 905  
vtss\_phy\_10g\_apc\_conf\_get  
    vtss\_phy\_10g\_api.h, 885  
vtss\_phy\_10g\_apc\_conf\_set  
    vtss\_phy\_10g\_api.h, 885  
vtss\_phy\_10g\_apc\_conf\_t, 234  
    op\_mode, 234  
    op\_mode\_flag, 235  
vtss\_phy\_10g\_apc\_restart  
    vtss\_phy\_10g\_api.h, 886  
vtss\_phy\_10g\_apc\_status\_get  
    vtss\_phy\_10g\_api.h, 886  
vtss\_phy\_10g\_apc\_status\_t, 235  
    freeze, 236  
    reset, 235  
vtss\_phy\_10g\_api.h  
    AMPLITUDE\_POINTS, 848  
    apc\_ib\_regulator\_t, 864  
    BOOLEAN\_STORAGE\_COUNT, 847  
    ckout\_sel\_, 871  
    ckout\_sel\_t, 861  
    clk\_mstr\_t, 865  
    ddr\_mode\_t, 865  
    oper\_mode\_t, 861  
    PHASE\_POINTS, 848  
    UNSIGNED\_STORAGE\_COUNT, 847  
    VTSS\_10G\_PHY\_GPIO\_MAL\_MAX, 849  
    VTSS\_10G\_PHY\_GPIO\_MAX, 849  
    VTSS\_PHY\_10G\_FEC\_FIXED\_CNT\_THRESH←  
        \_EV, 858  
    VTSS\_PHY\_10G\_FEC\_UNFIXED\_CNT\_THRESH←  
        \_SH\_EV, 857  
    VTSS\_PHY\_10G\_GPIO\_INT\_AGG0\_EV, 858  
    VTSS\_PHY\_10G\_GPIO\_INT\_AGG1\_EV, 859  
    VTSS\_PHY\_10G\_GPIO\_INT\_AGG2\_EV, 859  
    VTSS\_PHY\_10G\_GPIO\_INT\_AGG3\_EV, 859  
VTSS\_PHY\_10G\_HIGH\_BER\_EV, 850  
VTSS\_PHY\_10G\_HIGHBER\_EV, 858  
VTSS\_PHY\_10G\_HOST\_MAC\_LOCAL\_FAULT←  
    \_EV, 860  
VTSS\_PHY\_10G\_HOST\_MAC\_REMOTE\_FAU←  
    \_LT\_EV, 860  
VTSS\_PHY\_10G\_LINE\_MAC\_LOCAL\_FAULT←  
    \_EV, 860  
VTSS\_PHY\_10G\_LINE\_MAC\_REMOTE\_FAUL←  
    \_T\_EV, 860  
VTSS\_PHY\_10G\_LINK\_LOS\_EV, 849  
VTSS\_PHY\_10G\_LOPC\_EV, 850  
VTSS\_PHY\_10G\_MACSEC\_DISABLED, 848  
VTSS\_PHY\_10G\_MACSEC\_KEY\_128, 849  
VTSS\_PHY\_10G\_MODULE\_STAT\_EV, 850  
VTSS\_PHY\_10G\_ONE\_LINE\_ACTIVE, 848  
VTSS\_PHY\_10G\_PCS\_RECEIVE\_FAULT\_EV,  
    850  
VTSS\_PHY\_10G\_RX\_BLK\_DEC\_CNT\_THRES←  
    \_H\_EV, 857  
VTSS\_PHY\_10G\_RX\_CHAR\_DEC\_CNT\_THR←  
    \_ESH\_EV, 856  
VTSS\_PHY\_10G\_RX\_LINK\_STAT\_EV, 858  
VTSS\_PHY\_10G\_RX\_LOL\_EV, 849, 856  
VTSS\_PHY\_10G\_RX\_LOS\_EV, 856  
VTSS\_PHY\_10G\_RX\_SEQ\_CNT\_THRESH\_EV,  
    857  
VTSS\_PHY\_10G\_TIMESTAMP\_DISABLED, 848  
VTSS\_PHY\_10G\_TX\_BLK\_ENC\_CNT\_THRES←  
    \_H\_EV, 857  
VTSS\_PHY\_10G\_TX\_CHAR\_ENC\_CNT\_THRE←  
    \_SH\_EV, 856  
VTSS\_PHY\_10G\_TX\_LOL\_EV, 850, 856  
VTSS\_PHY\_10G\_TX\_SEQ\_CNT\_THRESH\_EV,  
    857  
VTSS\_PHY\_1G\_HOST\_AUTONEG\_RESTART←  
    \_EV, 859  
VTSS\_PHY\_1G\_LINE\_AUTONEG\_RESTART←  
    \_EV, 859  
VTSS\_PHY\_EWIS\_AISL\_EV, 852  
VTSS\_PHY\_EWIS\_AISP\_EV, 852  
VTSS\_PHY\_EWIS\_B1\_NZ\_EV, 854  
VTSS\_PHY\_EWIS\_B1\_THRESH\_EV, 855  
VTSS\_PHY\_EWIS\_B2\_NZ\_EV, 854  
VTSS\_PHY\_EWIS\_B2\_THRESH\_EV, 855  
VTSS\_PHY\_EWIS\_B3\_NZ\_EV, 854  
VTSS\_PHY\_EWIS\_B3\_THRESH\_EV, 855  
VTSS\_PHY\_EWIS\_FAIS\_EV, 851  
VTSS\_PHY\_EWIS\_FERDIP\_EV, 853  
VTSS\_PHY\_EWIS\_FEUNEQP\_EV, 853  
VTSS\_PHY\_EWIS\_FPLM\_EV, 851  
VTSS\_PHY\_EWIS\_LCDP\_EV, 852  
VTSS\_PHY\_EWIS\_LOF\_EV, 851  
VTSS\_PHY\_EWIS\_LOPP\_EV, 852  
VTSS\_PHY\_EWIS\_PLMP\_EV, 852  
VTSS\_PHY\_EWIS\_RDIL\_EV, 851  
VTSS\_PHY\_EWIS\_REIL\_EV, 853  
VTSS\_PHY\_EWIS\_REIL\_NZ\_EV, 854

VTSS\_PHY\_EWIS\_REIL\_THRESH\_EV, 855  
 VTSS\_PHY\_EWIS\_REIP\_EV, 853  
 VTSS\_PHY\_EWIS\_REIP\_NZ\_EV, 854  
 VTSS\_PHY\_EWIS\_REIP\_THRESH\_EV, 855  
 VTSS\_PHY\_EWIS\_SEF\_EV, 851  
 VTSS\_PHY\_EWIS\_UNEQP\_EV, 853  
 VTSS\_SLEWRATE\_120PS, 847  
 VTSS\_SLEWRATE\_25PS, 846  
 VTSS\_SLEWRATE\_35PS, 846  
 VTSS\_SLEWRATE\_55PS, 846  
 VTSS\_SLEWRATE\_70PS, 847  
 VTSS\_SLEWRATE\_INVALID, 847  
 vtss\_10g\_phy\_gpio\_t, 876  
 vtss\_32\_cntr\_t, 861  
 vtss\_channel\_t, 867  
 vtss\_ckout\_data\_sel\_t, 869  
 vtss\_gpio\_10g\_aggr\_intrpt\_t, 880  
 vtss\_gpio\_10g\_chan\_intrpt\_t, 879  
 vtss\_gpio\_10g\_gpio\_intr\_sgnl\_t, 877  
 vtss\_gpio\_10g\_input\_t, 881  
 vtss\_gpio\_10g\_led\_blink\_t, 882  
 vtss\_gpio\_10g\_led\_mode\_t, 881  
 vtss\_gpio\_10g\_no\_t, 861  
 vtss\_lb\_type\_t, 873  
 vtss\_phy\_10G\_is\_valid, 905  
 vtss\_phy\_10g\_apc\_conf\_get, 885  
 vtss\_phy\_10g\_apc\_conf\_set, 885  
 vtss\_phy\_10g\_apc\_restart, 886  
 vtss\_phy\_10g\_apc\_status\_get, 886  
 vtss\_phy\_10g\_auto\_failover\_event\_t, 875  
 vtss\_phy\_10g\_auto\_failover\_filter\_t, 875  
 vtss\_phy\_10g\_auto\_failover\_get, 907  
 vtss\_phy\_10g\_auto\_failover\_set, 907  
 vtss\_phy\_10g\_base\_host\_kr\_train\_aneg\_set, 897  
 vtss\_phy\_10g\_base\_kr\_conf\_get, 898  
 vtss\_phy\_10g\_base\_kr\_conf\_set, 898  
 vtss\_phy\_10g\_base\_kr\_host\_conf\_get, 899  
 vtss\_phy\_10g\_base\_kr\_host\_conf\_set, 899  
 vtss\_phy\_10g\_base\_kr\_train\_aneg\_get, 896  
 vtss\_phy\_10g\_base\_kr\_train\_aneg\_set, 897  
 vtss\_phy\_10g\_ckout\_conf\_set, 893  
 vtss\_phy\_10g\_ckout\_freq\_t, 868  
 vtss\_phy\_10g\_clause\_37\_control\_get, 902  
 vtss\_phy\_10g\_clause\_37\_control\_set, 903  
 vtss\_phy\_10g\_clause\_37\_remote\_fault\_t, 873  
 vtss\_phy\_10g\_clause\_37\_status\_get, 902  
 vtss\_phy\_10g\_clk\_sel\_t, 870  
 vtss\_phy\_10g\_cnt\_get, 904  
 vtss\_phy\_10g\_csr\_read, 922  
 vtss\_phy\_10g\_csr\_write, 922  
 vtss\_phy\_10g\_debug\_register\_dump, 896  
 vtss\_phy\_10g\_edc\_fw\_status\_get, 921  
 vtss\_phy\_10g\_event\_enable\_get, 918  
 vtss\_phy\_10g\_event\_enable\_set, 917  
 vtss\_phy\_10g\_event\_poll, 919  
 vtss\_phy\_10g\_event\_t, 861  
 vtss\_phy\_10g\_extended\_event\_enable\_get, 918  
 vtss\_phy\_10g\_extended\_event\_enable\_set, 920  
 vtss\_phy\_10g\_extended\_event\_poll, 920  
 vtss\_phy\_10g\_extnd\_event\_t, 861  
 vtss\_phy\_10g\_failover\_get, 906  
 vtss\_phy\_10g\_failover\_mode\_t, 874  
 vtss\_phy\_10g\_failover\_set, 906  
 vtss\_phy\_10g\_fc\_buffer\_reset, 921  
 vtss\_phy\_10g\_get\_user\_data, 926  
 vtss\_phy\_10g\_gpio\_mode\_get, 916  
 vtss\_phy\_10g\_gpio\_mode\_set, 916  
 vtss\_phy\_10g\_gpio\_read, 917  
 vtss\_phy\_10g\_gpio\_write, 917  
 vtss\_phy\_10g\_host\_clk\_conf\_set, 894  
 vtss\_phy\_10g\_host\_recvrd\_clk\_conf\_set, 895  
 vtss\_phy\_10g\_i2c\_read, 924  
 vtss\_phy\_10g\_i2c\_reset, 923  
 vtss\_phy\_10g\_i2c\_slave\_conf\_get, 925  
 vtss\_phy\_10g\_i2c\_slave\_conf\_set, 925  
 vtss\_phy\_10g\_i2c\_write, 924  
 vtss\_phy\_10g\_ib\_apc\_op\_mode\_t, 867  
 vtss\_phy\_10g\_ib\_conf\_get, 884  
 vtss\_phy\_10g\_ib\_conf\_set, 883  
 vtss\_phy\_10g\_ib\_status\_get, 884  
 vtss\_phy\_10g\_id\_get, 915  
 vtss\_phy\_10g\_init, 882  
 vtss\_phy\_10g\_jitter\_conf\_get, 887  
 vtss\_phy\_10g\_jitter\_conf\_set, 887  
 vtss\_phy\_10g\_jitter\_status\_get, 888  
 vtss\_phy\_10g\_kr\_status\_get, 900  
 vtss\_phy\_10g\_lane\_sync\_set, 895  
 vtss\_phy\_10g\_line\_clk\_conf\_set, 894  
 vtss\_phy\_10g\_line\_recvrd\_clk\_conf\_set, 895  
 vtss\_phy\_10g\_loopback\_get, 904  
 vtss\_phy\_10g\_loopback\_set, 903  
 vtss\_phy\_10g\_media\_t, 866  
 vtss\_phy\_10g\_mode\_get, 882  
 vtss\_phy\_10g\_mode\_set, 883  
 vtss\_phy\_10g\_ob\_status\_get, 900  
 vtss\_phy\_10g\_pcs\_prbs\_gen\_conf\_get, 909  
 vtss\_phy\_10g\_pcs\_prbs\_gen\_conf\_set, 909  
 vtss\_phy\_10g\_pcs\_prbs\_mon\_conf\_get, 910  
 vtss\_phy\_10g\_pcs\_prbs\_mon\_conf\_set, 910  
 vtss\_phy\_10g\_pcs\_prbs\_mon\_status\_get, 911  
 vtss\_phy\_10g\_pcs\_status\_get, 919  
 vtss\_phy\_10g\_pkt\_gen\_conf, 914  
 vtss\_phy\_10g\_pkt\_mon\_conf, 914  
 vtss\_phy\_10g\_pkt\_mon\_counters\_get, 915  
 vtss\_phy\_10g\_pkt\_mon\_rst\_t, 876  
 vtss\_phy\_10g\_poll\_1sec, 920  
 vtss\_phy\_10g\_power\_get, 905  
 vtss\_phy\_10g\_power\_set, 905  
 vtss\_phy\_10g\_power\_t, 874  
 vtss\_phy\_10g\_prbs\_gen\_conf, 911  
 vtss\_phy\_10g\_prbs\_gen\_conf\_get, 912  
 vtss\_phy\_10g\_prbs\_mon\_conf, 912  
 vtss\_phy\_10g\_prbs\_mon\_conf\_get, 913  
 vtss\_phy\_10g\_prbs\_mon\_status\_get, 913  
 vtss\_phy\_10g\_recvrd\_clk\_sel\_t, 871  
 vtss\_phy\_10g\_reset, 901

vtss\_phy\_10g\_rx\_macro\_t, 872  
vtss\_phy\_10g\_rxckout\_get, 890  
vtss\_phy\_10g\_rxckout\_set, 890  
vtss\_phy\_10g\_sckout\_conf\_set, 893  
vtss\_phy\_10g\_sckout\_freq\_t, 872  
vtss\_phy\_10g\_serdes\_status\_get, 901  
vtss\_phy\_10g\_sgmii\_mode\_set, 923  
vtss\_phy\_10g\_squelch\_src\_t, 869  
vtss\_phy\_10g\_srefclk\_conf\_get, 892  
vtss\_phy\_10g\_srefclk\_conf\_set, 892  
vtss\_phy\_10g\_srefclk\_freq\_t, 868  
vtss\_phy\_10g\_status\_get, 901  
vtss\_phy\_10g\_syncce\_clkout\_get, 888  
vtss\_phy\_10g\_syncce\_clkout\_set, 889  
vtss\_phy\_10g\_tx\_macro\_t, 872  
vtss\_phy\_10g\_txckout\_get, 891  
vtss\_phy\_10g\_txckout\_set, 891  
vtss\_phy\_10g\_vscope\_conf\_get, 908  
vtss\_phy\_10g\_vscope\_conf\_set, 908  
vtss\_phy\_10g\_vscope\_scan\_status\_get, 908  
vtss\_phy\_10g\_vscope\_scan\_t, 875  
vtss\_phy\_10g\_xfp\_clkout\_get, 889  
vtss\_phy\_10g\_xfp\_clkout\_set, 890  
vtss\_phy\_6g\_link\_partner\_distance\_t, 866  
vtss\_phy\_interface\_mode, 863  
vtss\_phy\_warm\_start\_10g\_failed\_get, 923  
vtss\_recvrd\_clkout\_t, 867  
vtss\_recvrd\_t, 863  
vtss\_recvrdclk\_cdr\_div\_t, 863  
vtss\_rptr\_rate\_t, 865  
vtss\_srefclk\_div\_t, 864  
vtss\_wref\_clk\_div\_t, 864  
vtss\_wrefclk\_t, 862  
vtss\_phy\_10g\_auto\_failover\_conf\_t, 236  
a\_gpio, 238  
channel\_id, 237  
enable, 238  
evnt, 237  
filter, 238  
filtr\_val, 238  
is\_host\_side, 237  
port\_no, 237  
trig\_ch\_id, 237  
v\_gpio, 237  
vtss\_phy\_10g\_auto\_failover\_event\_t  
  vtss\_phy\_10g\_api.h, 875  
vtss\_phy\_10g\_auto\_failover\_filter\_t  
  vtss\_phy\_10g\_api.h, 875  
vtss\_phy\_10g\_auto\_failover\_get  
  vtss\_phy\_10g\_api.h, 907  
vtss\_phy\_10g\_auto\_failover\_set  
  vtss\_phy\_10g\_api.h, 907  
vtss\_phy\_10g\_base\_host\_kr\_train\_aneg\_set  
  vtss\_phy\_10g\_api.h, 897  
vtss\_phy\_10g\_base\_kr\_autoneg\_t, 239  
an\_enable, 239  
an\_reset, 239  
an\_restart, 239  
vtss\_phy\_10g\_base\_kr\_conf\_get  
  vtss\_phy\_10g\_api.h, 898  
vtss\_phy\_10g\_base\_kr\_conf\_set  
  vtss\_phy\_10g\_api.h, 898  
vtss\_phy\_10g\_base\_kr\_conf\_t, 240  
  ampl, 241  
  c0, 240  
  c1, 241  
  cm1, 240  
  en\_ob, 241  
  ser\_inv, 241  
  slewrate, 241  
vtss\_phy\_10g\_base\_kr\_host\_conf\_get  
  vtss\_phy\_10g\_api.h, 899  
vtss\_phy\_10g\_base\_kr\_host\_conf\_set  
  vtss\_phy\_10g\_api.h, 899  
vtss\_phy\_10g\_base\_kr\_ld\_adv\_abil\_t, 242  
  adv\_10g, 242  
  adv\_1g, 242  
  fec\_abil, 243  
  fec\_req, 243  
vtss\_phy\_10g\_base\_kr\_status\_t, 243  
  aneg, 244  
  fec, 244  
  train, 244  
vtss\_phy\_10g\_base\_kr\_train\_aneg\_get  
  vtss\_phy\_10g\_api.h, 896  
vtss\_phy\_10g\_base\_kr\_train\_aneg\_set  
  vtss\_phy\_10g\_api.h, 897  
vtss\_phy\_10g\_base\_kr\_train\_aneg\_t, 244  
  autoneg, 245  
  host\_kr, 245  
  ld\_abil, 245  
  line\_kr, 245  
  training, 245  
vtss\_phy\_10g\_base\_kr\_training\_t, 246  
  enable, 246  
  ld\_pre\_init, 247  
  trmthd\_c0, 247  
  trmthd\_cm, 247  
  trmthd\_cp, 246  
vtss\_phy\_10g\_ckout\_conf\_set  
  vtss\_phy\_10g\_api.h, 893  
vtss\_phy\_10g\_ckout\_conf\_t, 247  
  ckout\_sel, 249  
  enable, 249  
  freq, 248  
  mode, 248  
  squelch\_inv, 248  
  src, 248  
vtss\_phy\_10g\_ckout\_freq\_t  
  vtss\_phy\_10g\_api.h, 868  
vtss\_phy\_10g\_clause\_37\_adv\_t, 249  
  acknowledge, 251  
  asymmetric\_pause, 250  
  fdx, 250  
  hdx, 250  
  next\_page, 251

remote\_fault, 250  
 symmetric\_pause, 250  
 vtss\_phy\_10g\_clause\_37\_cmn\_status\_t, 251  
 host, 252  
 line, 252  
 vtss\_phy\_10g\_clause\_37\_control\_get  
     vtss\_phy\_10g\_api.h, 902  
 vtss\_phy\_10g\_clause\_37\_control\_set  
     vtss\_phy\_10g\_api.h, 903  
 vtss\_phy\_10g\_clause\_37\_control\_t, 252  
     advertisement, 253  
     enable, 253  
     enable\_pass\_thru, 253  
     host, 253  
     l\_h, 253  
     line, 253  
 vtss\_phy\_10g\_clause\_37\_remote\_fault\_t  
     vtss\_phy\_10g\_api.h, 873  
 vtss\_phy\_10g\_clause\_37\_status\_get  
     vtss\_phy\_10g\_api.h, 902  
 vtss\_phy\_10g\_clause\_37\_status\_t, 254  
     autoneg, 255  
     complete, 254  
     link, 254  
     partner\_advertisement, 255  
 vtss\_phy\_10g\_clk\_sel\_t  
     vtss\_phy\_10g\_api.h, 870  
 vtss\_phy\_10g\_clk\_src\_t, 255  
     is\_high\_amp, 256  
 vtss\_phy\_10g\_cnt\_get  
     vtss\_phy\_10g\_api.h, 904  
 vtss\_phy\_10g\_cnt\_t, 256  
     pcs, 256  
 vtss\_phy\_10g\_csr\_read  
     vtss\_phy\_10g\_api.h, 922  
 vtss\_phy\_10g\_csr\_write  
     vtss\_phy\_10g\_api.h, 922  
 vtss\_phy\_10g\_debug\_register\_dump  
     vtss\_phy\_10g\_api.h, 896  
 vtss\_phy\_10g\_edc\_fw\_status\_get  
     vtss\_phy\_10g\_api.h, 921  
 vtss\_phy\_10g\_event\_enable\_get  
     vtss\_phy\_10g\_api.h, 918  
 vtss\_phy\_10g\_event\_enable\_set  
     vtss\_phy\_10g\_api.h, 917  
 vtss\_phy\_10g\_event\_poll  
     vtss\_phy\_10g\_api.h, 919  
 vtss\_phy\_10g\_event\_t  
     vtss\_phy\_10g\_api.h, 861  
 vtss\_phy\_10g\_extended\_event\_enable\_get  
     vtss\_phy\_10g\_api.h, 918  
 vtss\_phy\_10g\_extended\_event\_enable\_set  
     vtss\_phy\_10g\_api.h, 920  
 vtss\_phy\_10g\_extended\_event\_poll  
     vtss\_phy\_10g\_api.h, 920  
 vtss\_phy\_10g\_extnd\_event\_t  
     vtss\_phy\_10g\_api.h, 861  
 vtss\_phy\_10g\_failover\_get  
     vtss\_phy\_10g\_api.h, 906  
     vtss\_phy\_10g\_failover\_mode\_t  
         vtss\_phy\_10g\_api.h, 874  
     vtss\_phy\_10g\_failover\_set  
         vtss\_phy\_10g\_api.h, 906  
     vtss\_phy\_10g\_fc\_buffer\_reset  
         vtss\_phy\_10g\_api.h, 921  
 vtss\_phy\_10g\_fifo\_sync\_t, 257  
     bypass\_in\_api, 257  
     pr, 257  
     skip\_rev\_check, 257  
 vtss\_phy\_10g\_fw\_status\_t, 258  
     edc\_fw\_api\_load, 259  
     edc\_fw\_chksum, 258  
     edc\_fw\_rev, 258  
     icpu\_activity, 259  
 vtss\_phy\_10g\_get\_user\_data  
     vtss\_phy\_10g\_api.h, 926  
 vtss\_phy\_10g\_gpio\_mode\_get  
     vtss\_phy\_10g\_api.h, 916  
 vtss\_phy\_10g\_gpio\_mode\_set  
     vtss\_phy\_10g\_api.h, 916  
 vtss\_phy\_10g\_gpio\_read  
     vtss\_phy\_10g\_api.h, 917  
 vtss\_phy\_10g\_gpio\_write  
     vtss\_phy\_10g\_api.h, 917  
 vtss\_phy\_10g\_host\_clk\_conf\_set  
     vtss\_phy\_10g\_api.h, 894  
 vtss\_phy\_10g\_host\_clk\_conf\_t, 259  
     clk\_sel\_no, 260  
     mode, 260  
     recvrd\_clk\_sel, 260  
 vtss\_phy\_10g\_host\_recvrd\_clk\_conf\_set  
     vtss\_phy\_10g\_api.h, 895  
 vtss\_phy\_10g\_i2c\_read  
     vtss\_phy\_10g\_api.h, 924  
 vtss\_phy\_10g\_i2c\_reset  
     vtss\_phy\_10g\_api.h, 923  
 vtss\_phy\_10g\_i2c\_slave\_conf\_get  
     vtss\_phy\_10g\_api.h, 925  
 vtss\_phy\_10g\_i2c\_slave\_conf\_set  
     vtss\_phy\_10g\_api.h, 925  
 vtss\_phy\_10g\_i2c\_slave\_conf\_t, 260  
     prescale, 261  
     slave\_id, 261  
 vtss\_phy\_10g\_i2c\_write  
     vtss\_phy\_10g\_api.h, 924  
 vtss\_phy\_10g\_ib\_apc\_op\_mode\_t  
     vtss\_phy\_10g\_api.h, 867  
 vtss\_phy\_10g\_ib\_conf\_get  
     vtss\_phy\_10g\_api.h, 884  
 vtss\_phy\_10g\_ib\_conf\_set  
     vtss\_phy\_10g\_api.h, 883  
 vtss\_phy\_10g\_ib\_conf\_t, 261  
     agc, 263  
     apc\_bit\_mask, 265  
     c, 263  
     config\_bit\_mask, 265

dfe1, 263  
dfe2, 263  
dfe3, 264  
dfe4, 264  
freeze\_bit\_mask, 265  
gain, 262  
gainadj, 262  
is\_host, 265  
I, 263  
Id, 264  
offs, 262  
prbs, 264  
prbs\_inv, 264  
vtss\_phy\_10g\_ib\_status\_get  
  vtss\_phy\_10g\_api.h, 884  
vtss\_phy\_10g\_ib\_status\_t, 266  
  bit\_errors, 266  
  ib\_conf, 266  
  sig\_det, 266  
vtss\_phy\_10g\_ib\_storage\_t, 267  
  ib\_storage, 267  
  ib\_storage\_bool, 267  
vtss\_phy\_10g\_id\_get  
  vtss\_phy\_10g\_api.h, 915  
vtss\_phy\_10g\_id\_t, 268  
  channel\_id, 269  
  device\_feature\_status, 269  
  family, 269  
  part\_number, 268  
  phy\_api\_base\_no, 269  
  revision, 268  
  type, 269  
vtss\_phy\_10g\_init  
  vtss\_phy\_10g\_api.h, 882  
vtss\_phy\_10g\_init\_parm\_t, 270  
  channel\_conf, 270  
vtss\_phy\_10g\_jitter\_conf\_get  
  vtss\_phy\_10g\_api.h, 887  
vtss\_phy\_10g\_jitter\_conf\_set  
  vtss\_phy\_10g\_api.h, 887  
vtss\_phy\_10g\_jitter\_conf\_t, 271  
  incr\_levn, 271  
  levn, 271  
  vtail, 271  
vtss\_phy\_10g\_jitter\_status\_get  
  vtss\_phy\_10g\_api.h, 888  
vtss\_phy\_10g\_kr\_status\_aneg\_t, 272  
  active, 272  
  block\_lock, 274  
  complete, 272  
  fec\_enable, 273  
  lp\_aneg\_able, 274  
  request\_10g, 273  
  request\_1g, 273  
  request\_fec\_change, 273  
  sm, 273  
vtss\_phy\_10g\_kr\_status\_fec\_t, 274  
  corrected\_block\_cnt, 275  
enable, 275  
uncorrected\_block\_cnt, 275  
vtss\_phy\_10g\_kr\_status\_get  
  vtss\_phy\_10g\_api.h, 900  
vtss\_phy\_10g\_kr\_status\_train\_t, 275  
  c0\_ob\_tap\_result, 276  
  cm\_ob\_tap\_result, 276  
  complete, 276  
  cp\_ob\_tap\_result, 276  
vtss\_phy\_10g\_lane\_sync\_conf\_t, 277  
  enable, 277  
  rx\_ch, 278  
  rx\_macro, 277  
  tx\_ch, 278  
  tx\_macro, 277  
vtss\_phy\_10g\_lane\_sync\_set  
  vtss\_phy\_10g\_api.h, 895  
vtss\_phy\_10g\_line\_clk\_conf\_set  
  vtss\_phy\_10g\_api.h, 894  
vtss\_phy\_10g\_line\_clk\_conf\_t, 278  
  clk\_sel\_no, 279  
  mode, 279  
  recvrd\_clk\_sel, 279  
vtss\_phy\_10g\_line\_recvrd\_clk\_conf\_set  
  vtss\_phy\_10g\_api.h, 895  
vtss\_phy\_10g\_loopback\_get  
  vtss\_phy\_10g\_api.h, 904  
vtss\_phy\_10g\_loopback\_set  
  vtss\_phy\_10g\_api.h, 903  
vtss\_phy\_10g\_loopback\_t, 279  
  enable, 280  
  lb\_type, 280  
vtss\_phy\_10g\_media\_t  
  vtss\_phy\_10g\_api.h, 866  
vtss\_phy\_10g\_mode\_get  
  vtss\_phy\_10g\_api.h, 882  
vtss\_phy\_10g\_mode\_set  
  vtss\_phy\_10g\_api.h, 883  
vtss\_phy\_10g\_mode\_t, 280  
  apc\_eqz\_offs\_par\_cfg, 287  
  apc\_host\_eqz\_id\_ctrl, 287  
  apc\_host\_id\_ctrl, 286  
  apc\_ib\_regulator, 288  
  apc\_line\_eqz\_id\_ctrl, 287  
  apc\_line\_id\_ctrl, 286  
  apc\_offs\_ctrl, 285  
  cfg0, 286  
  channel\_id, 283  
  d\_filter, 286  
  ddr\_mode, 289  
  dig\_offset\_reg, 285  
  edc\_fw\_load, 284  
  enable\_pass\_thru, 292  
  h\_apc\_conf, 291  
  h\_clk\_src, 290  
  h\_ib\_conf, 291  
  h\_media, 291  
  h\_offset\_guard, 288

high\_input\_gain, 283  
 hl\_clk\_synth, 283  
 ib\_conf, 285  
 ib\_ini\_lp, 286  
 ib\_max\_lp, 287  
 ib\_min\_lp, 287  
 interface, 282  
 is\_host\_wan, 290  
 is\_init, 292  
 l\_apc\_conf, 292  
 l\_clk\_src, 290  
 l\_ib\_conf, 291  
 l\_media, 291  
 l\_offset\_guard, 288  
 link\_6g\_distance, 290  
 lref\_for\_host, 290  
 master, 289  
 ob\_conf, 285  
 oper\_mode, 282  
 pma\_txratecontrol, 288  
 polarity, 289  
 rate, 289  
 rcvrd\_clk, 284  
 rcvrd\_clk\_div, 284  
 sd6g\_calib\_done, 292  
 serdes\_conf, 288  
 sref\_clk\_div, 284  
 use\_conf, 285  
 venice\_rev\_a\_los\_detection\_workaround, 289  
 wref\_clk\_div, 284  
 wrefclk, 282  
 xaui\_lane\_flip, 283  
 xfi\_pol\_invert, 283  
 vtss\_phy\_10g\_ob\_status\_get  
     vtss\_phy\_10g\_api.h, 900  
 vtss\_phy\_10g\_ob\_status\_t, 293  
     c\_ctrl, 293  
     d\_fltr, 294  
     is\_host, 295  
     levn, 294  
     r\_ctrl, 293  
     slew, 293  
     v3, 294  
     v4, 294  
     v5, 295  
     vp, 294  
 vtss\_phy\_10g\_pcs\_prbs\_gen\_conf\_get  
     vtss\_phy\_10g\_api.h, 909  
 vtss\_phy\_10g\_pcs\_prbs\_gen\_conf\_set  
     vtss\_phy\_10g\_api.h, 909  
 vtss\_phy\_10g\_pcs\_prbs\_gen\_conf\_t, 295  
     prbs\_gen, 296  
 vtss\_phy\_10g\_pcs\_prbs\_mon\_conf\_get  
     vtss\_phy\_10g\_api.h, 910  
 vtss\_phy\_10g\_pcs\_prbs\_mon\_conf\_set  
     vtss\_phy\_10g\_api.h, 910  
 vtss\_phy\_10g\_pcs\_prbs\_mon\_conf\_t, 296  
     error\_counter, 296  
             prbs\_mon, 296  
             vtss\_phy\_10g\_pcs\_prbs\_mon\_status\_get  
                 vtss\_phy\_10g\_api.h, 911  
             vtss\_phy\_10g\_pcs\_status\_get  
                 vtss\_phy\_10g\_api.h, 919  
             vtss\_phy\_10g\_pkt\_gen\_conf  
                 vtss\_phy\_10g\_api.h, 914  
             vtss\_phy\_10g\_pkt\_gen\_conf\_t, 297  
                 dmac, 299  
                 enable, 297  
                 etype, 298  
                 frame\_single, 298  
                 frames, 298  
                 ingress, 298  
                 ipg\_len, 299  
                 pkt\_len, 299  
                 ptp, 298  
                 ptp\_ts\_ns, 300  
                 ptp\_ts\_sec, 299  
                 smac, 299  
                 srade, 300  
             vtss\_phy\_10g\_pkt\_mon\_conf  
                 vtss\_phy\_10g\_api.h, 914  
 vtss\_phy\_10g\_pkt\_mon\_conf\_t, 300  
     bad\_crc, 301  
     ber, 302  
     enable, 301  
     frag, 302  
     good\_crc, 301  
     lfault, 302  
     reset, 301  
     update, 301  
 vtss\_phy\_10g\_pkt\_mon\_counters\_get  
     vtss\_phy\_10g\_api.h, 915  
 vtss\_phy\_10g\_pkt\_mon\_rst\_t  
     vtss\_phy\_10g\_api.h, 876  
 vtss\_phy\_10g\_polarity\_inv\_t, 302  
     host\_rx, 303  
     host\_tx, 303  
     line\_rx, 303  
     line\_tx, 303  
 vtss\_phy\_10g\_poll\_1sec  
     vtss\_phy\_10g\_api.h, 920  
 vtss\_phy\_10g\_power\_get  
     vtss\_phy\_10g\_api.h, 905  
 vtss\_phy\_10g\_power\_set  
     vtss\_phy\_10g\_api.h, 905  
 vtss\_phy\_10g\_power\_t  
     vtss\_phy\_10g\_api.h, 874  
 vtss\_phy\_10g\_prbs\_gen\_conf  
     vtss\_phy\_10g\_api.h, 911  
 vtss\_phy\_10g\_prbs\_gen\_conf\_get  
     vtss\_phy\_10g\_api.h, 912  
 vtss\_phy\_10g\_prbs\_gen\_conf\_t, 304  
     enable, 304  
     line, 305  
     prbsn\_tx\_io, 305  
     prbsn\_tx\_iw, 305

prbsn\_tx\_sel, 304  
vtss\_phy\_10g\_prbs\_mon\_conf  
  vtss\_phy\_10g\_api.h, 912  
vtss\_phy\_10g\_prbs\_mon\_conf\_get  
  vtss\_phy\_10g\_api.h, 913  
vtss\_phy\_10g\_prbs\_mon\_conf\_t, 305  
  active, 309  
  bist\_mode, 308  
  des\_interface\_width, 307  
  enable, 306  
  error\_states, 307  
  error\_status, 308  
  incomplete, 309  
  instable, 309  
  line, 306  
  main\_status, 308  
  max\_bist\_frames, 306  
  no\_of\_errors, 307  
  no\_sync, 309  
  PRBS\_status, 308  
  prbs\_check\_input\_invert, 307  
  prbsn\_sel, 307  
  stuck\_at\_01, 309  
  stuck\_at\_par, 308  
vtss\_phy\_10g\_prbs\_mon\_status\_get  
  vtss\_phy\_10g\_api.h, 913  
vtss\_phy\_10g\_recvrd\_clk\_sel\_t  
  vtss\_phy\_10g\_api.h, 871  
vtss\_phy\_10g\_reset  
  vtss\_phy\_10g\_api.h, 901  
vtss\_phy\_10g\_rx\_macro\_t  
  vtss\_phy\_10g\_api.h, 872  
vtss\_phy\_10g\_rxckout\_conf\_t, 310  
  mode, 310  
  squench\_on\_lopc, 311  
  squench\_on\_pcs\_fault, 310  
vtss\_phy\_10g\_rxckout\_get  
  vtss\_phy\_10g\_api.h, 890  
vtss\_phy\_10g\_rxckout\_set  
  vtss\_phy\_10g\_api.h, 890  
vtss\_phy\_10g\_sckout\_conf\_set  
  vtss\_phy\_10g\_api.h, 893  
vtss\_phy\_10g\_sckout\_conf\_t, 311  
  enable, 312  
  freq, 312  
  mode, 311  
  squench\_inv, 312  
  src, 312  
vtss\_phy\_10g\_sckout\_freq\_t  
  vtss\_phy\_10g\_api.h, 872  
vtss\_phy\_10g\_serdes\_status\_get  
  vtss\_phy\_10g\_api.h, 901  
vtss\_phy\_10g\_serdes\_status\_t, 313  
  h\_pcs, 319  
  h\_pll5g\_fsm\_lock, 314  
  h\_pll5g\_fsm\_stat, 314  
  h\_pll5g\_gain, 314  
  h\_pll5g\_lock\_status, 314  
  h\_pma, 318  
  h\_rx\_rcpll\_fsm\_status, 316  
  h\_rx\_rcpll\_lock\_status, 315  
  h\_rx\_rcpll\_range, 315  
  h\_rx\_rcpll\_vco\_load, 316  
  h\_tx\_rcpll\_fsm\_status, 317  
  h\_tx\_rcpll\_lock\_status, 317  
  h\_tx\_rcpll\_range, 317  
  h\_tx\_rcpll\_vco\_load, 317  
  l\_pcs, 319  
  l\_pll5g\_fsm\_lock, 315  
  l\_pll5g\_fsm\_stat, 315  
  l\_pll5g\_gain, 315  
  l\_pll5g\_lock\_status, 314  
  l\_pma, 319  
  l\_rx\_rcpll\_fsm\_status, 317  
  l\_rx\_rcpll\_lock\_status, 316  
  l\_rx\_rcpll\_range, 316  
  l\_rx\_rcpll\_vco\_load, 316  
  l\_tx\_rcpll\_fsm\_status, 318  
  l\_tx\_rcpll\_lock\_status, 318  
  l\_tx\_rcpll\_range, 318  
  l\_tx\_rcpll\_vco\_load, 318  
  rcomp, 313  
  wis, 319  
vtss\_phy\_10g\_sgmii\_mode\_set  
  vtss\_phy\_10g\_api.h, 923  
vtss\_phy\_10g\_squelch\_src\_t  
  vtss\_phy\_10g\_api.h, 869  
vtss\_phy\_10g\_srefclk\_conf\_get  
  vtss\_phy\_10g\_api.h, 892  
vtss\_phy\_10g\_srefclk\_conf\_set  
  vtss\_phy\_10g\_api.h, 892  
vtss\_phy\_10g\_srefclk\_freq\_t  
  vtss\_phy\_10g\_api.h, 868  
vtss\_phy\_10g\_srefclk\_mode\_t, 320  
  enable, 320  
  freq, 320  
vtss\_phy\_10g\_status\_get  
  vtss\_phy\_10g\_api.h, 901  
vtss\_phy\_10g\_status\_t, 320  
  block\_lock, 323  
  hpcs, 322  
  hpcs\_1g, 322  
  hpma, 321  
  lopc\_stat, 323  
  lpcs\_1g, 322  
  pcs, 321  
  pma, 321  
  status, 322  
  wis, 321  
  xs, 322  
vtss\_phy\_10g\_sync\_clkout\_get  
  vtss\_phy\_10g\_api.h, 888  
vtss\_phy\_10g\_sync\_clkout\_set  
  vtss\_phy\_10g\_api.h, 889  
vtss\_phy\_10g\_timestamp\_val\_t, 323  
  timestamp, 324

vtss\_phy\_10g\_tx\_macro\_t  
     vtss\_phy\_10g\_api.h, 872  
 vtss\_phy\_10g\_txckout\_conf\_t, 324  
     mode, 324  
 vtss\_phy\_10g\_txckout\_get  
     vtss\_phy\_10g\_api.h, 891  
 vtss\_phy\_10g\_txckout\_set  
     vtss\_phy\_10g\_api.h, 891  
 vtss\_phy\_10g\_vscope\_conf\_get  
     vtss\_phy\_10g\_api.h, 908  
 vtss\_phy\_10g\_vscope\_conf\_set  
     vtss\_phy\_10g\_api.h, 908  
 vtss\_phy\_10g\_vscope\_conf\_t, 325  
     enable, 325  
     error\_thres, 325  
     line, 325  
     scan\_type, 325  
 vtss\_phy\_10g\_vscope\_scan\_conf\_t, 326  
     ber, 328  
     line, 326  
     x\_count, 327  
     x\_incr, 327  
     x\_start, 326  
     y\_count, 327  
     y\_incr, 327  
     y\_start, 327  
 vtss\_phy\_10g\_vscope\_scan\_status\_get  
     vtss\_phy\_10g\_api.h, 908  
 vtss\_phy\_10g\_vscope\_scan\_status\_t, 328  
     amp\_range, 329  
     error\_free\_x, 329  
     error\_free\_y, 329  
     errors, 329  
     scan\_conf, 328  
 vtss\_phy\_10g\_vscope\_scan\_t  
     vtss\_phy\_10g\_api.h, 875  
 vtss\_phy\_10g\_xfp\_clkout\_get  
     vtss\_phy\_10g\_api.h, 889  
 vtss\_phy\_10g\_xfp\_clkout\_set  
     vtss\_phy\_10g\_api.h, 890  
 vtss\_phy\_1588\_csr\_reg\_read  
     vtss\_phy\_ts\_api.h, 1070  
 vtss\_phy\_1588\_csr\_reg\_write  
     vtss\_phy\_ts\_api.h, 1069  
 vtss\_phy\_1588\_debug\_reg\_read  
     vtss\_phy\_ts\_api.h, 1073  
 vtss\_phy\_1g\_ts\_fifo\_sync  
     vtss\_phy\_ts\_api.h, 1073  
 vtss\_phy\_6g\_link\_partner\_distance\_t  
     vtss\_phy\_10g\_api.h, 866  
 vtss\_phy\_aneg\_t, 330  
     asymmetric\_pause, 331  
     speed\_100m\_fdx, 331  
     speed\_100m\_hdx, 330  
     speed\_10m\_fdx, 330  
     speed\_10m\_hdx, 330  
     speed\_1g\_fdx, 331  
     speed\_1g\_hdx, 331  
     symmetric\_pause, 331  
     tx\_remote\_fault, 332  
 vtss\_phy\_api.h  
     lb\_type, 956  
     MAX\_CFG\_BUF\_SIZE, 936  
     MAX\_STAT\_BUF\_SIZE, 936  
     MAX\_WOL\_MAC\_ADDR\_SIZE, 947  
     MAX\_WOL\_PASSWD\_SIZE, 947  
     MIN\_WOL\_PASSWD\_SIZE, 947  
     rgmii\_skew\_delay\_psec\_t, 950  
     VTSS\_PHY\_ACTIPHY\_PWR, 937  
     VTSS\_PHY\_LINK\_AMS\_EV, 942  
     VTSS\_PHY\_LINK\_AUTO\_NEG\_COMPLETE\_EV, 942  
     VTSS\_PHY\_LINK\_AUTO\_NEG\_ERROR\_EV, 942  
     VTSS\_PHY\_LINK\_DOWN\_PWR, 937  
     VTSS\_PHY\_LINK\_EXT\_EEE\_LINKFAIL\_EV, 945  
     VTSS\_PHY\_LINK\_EXT\_EEE\_WAIT\_RX\_TQ\_EV, 945  
     VTSS\_PHY\_LINK\_EXT\_EEE\_WAIT\_TS\_EV, 945  
     VTSS\_PHY\_LINK\_EXT\_EEE\_WAKE\_ERR\_EV, 945  
     VTSS\_PHY\_LINK\_EXT\_MACSEC\_EGRESS\_EV, 946  
     VTSS\_PHY\_LINK\_EXT\_MACSEC\_FC\_BUFF←EV, 946  
     VTSS\_PHY\_LINK\_EXT\_MACSEC\_HOST\_MAC←EV, 946  
     VTSS\_PHY\_LINK\_EXT\_MACSEC\_INGRESS←EV, 946  
     VTSS\_PHY\_LINK\_EXT\_MACSEC\_LINE\_MAC←EV, 946  
     VTSS\_PHY\_LINK\_EXT\_MEM\_INT\_RING\_EV, 947  
     VTSS\_PHY\_LINK\_EXT\_RR\_SW\_COMPL\_EV, 945  
     VTSS\_PHY\_LINK\_EXTENDED\_REG\_INT\_EV, 944  
     VTSS\_PHY\_LINK\_FALSE\_CARRIER\_INT\_EV, 943  
     VTSS\_PHY\_LINK\_FDX\_STATE\_CHANGE\_EV, 942  
     VTSS\_PHY\_LINK\_FFAIL\_EV, 941  
     VTSS\_PHY\_LINK\_INLINE\_POW\_DEV\_DETECT←T\_EV, 943  
     VTSS\_PHY\_LINK\_LINK\_SPEED\_DS\_DETECT←EV, 944  
     VTSS\_PHY\_LINK\_LOS\_EV, 941  
     VTSS\_PHY\_LINK\_MASTER\_SLAVE\_RES\_ER←R\_EV, 944  
     VTSS\_PHY\_LINK\_RX\_ER\_INT\_EV, 944  
     VTSS\_PHY\_LINK\_RX\_FIFO\_OVERFLOW\_INT←EV, 943  
     VTSS\_PHY\_LINK\_SPEED\_STATE\_CHANGE←EV, 942  
     VTSS\_PHY\_LINK\_SYMBOL\_ERR\_INT\_EV, 943  
     VTSS\_PHY\_LINK\_TX\_FIFO\_OVERFLOW\_INT←EV, 943

VTSS\_PHY\_LINK\_UP\_FULL\_PWR, 937  
VTSS\_PHY\_LINK\_WAKE\_ON\_LAN\_INT\_EV, 944  
VTSS\_PHY\_PAGE\_0x2DAF, 940  
VTSS\_PHY\_PAGE\_1588, 939  
VTSS\_PHY\_PAGE\_EXTENDED\_2, 938  
VTSS\_PHY\_PAGE\_EXTENDED\_3, 939  
VTSS\_PHY\_PAGE\_EXTENDED\_4, 939  
VTSS\_PHY\_PAGE\_EXTENDED, 938  
VTSS\_PHY\_PAGE\_GPIO, 939  
VTSS\_PHY\_PAGE\_MACSEC, 939  
VTSS\_PHY\_PAGE\_STANDARD, 938  
VTSS\_PHY\_PAGE\_TEST, 940  
VTSS\_PHY\_PAGE\_TR, 940  
VTSS\_PHY\_POWER\_ACTIPHY\_BIT, 936  
VTSS\_PHY\_POWER\_DYNAMIC\_BIT, 937  
VTSS\_PHY\_RECov\_CLK1, 937  
VTSS\_PHY\_RECov\_CLK2, 938  
VTSS\_PHY\_RECov\_CLK\_NUM, 938  
VTSS\_PHY\_REG\_EXTENDED, 940  
VTSS\_PHY\_REG\_GPIO, 941  
VTSS\_PHY\_REG\_STANDARD, 940  
VTSS\_PHY\_REG\_TEST, 941  
VTSS\_PHY\_REG\_TR, 941  
vga\_adc\_debug, 976  
vtss\_fefi\_mode\_t, 957  
vtss\_phy\_atom12\_patch\_settings\_get, 986  
vtss\_phy\_cfg\_ib\_cterm, 986  
vtss\_phy\_cfg\_ob\_post0, 985  
vtss\_phy\_chip\_temp\_get, 959  
vtss\_phy\_chip\_temp\_init, 960  
vtss\_phy\_cl37\_lp\_abil\_get, 961  
vtss\_phy\_clk\_source\_t, 955  
vtss\_phy\_clk\_squelch, 955  
vtss\_phy\_clock\_conf\_get, 965  
vtss\_phy\_clock\_conf\_set, 965  
vtss\_phy\_coma\_mode\_disable, 975  
vtss\_phy\_coma\_mode\_enable, 976  
vtss\_phy\_conf\_1g\_get, 962  
vtss\_phy\_conf\_1g\_set, 963  
vtss\_phy\_conf\_get, 960  
vtss\_phy\_conf\_set, 961  
vtss\_phy\_csr\_rd, 981  
vtss\_phy\_csr\_wr, 981  
vtss\_phy\_debug\_phyinfo\_print, 989  
vtss\_phy\_debug\_regdump\_print, 991  
vtss\_phy\_debug\_register\_dump, 989  
vtss\_phy\_debug\_stat\_print, 988  
vtss\_phy\_detect\_base\_ports, 990  
vtss\_phy\_do\_page\_chk\_get, 984  
vtss\_phy\_do\_page\_chk\_set, 983  
vtss\_phy\_eee\_conf\_get, 977  
vtss\_phy\_eee\_conf\_set, 978  
vtss\_phy\_eee\_ena, 977  
vtss\_phy\_eee\_link\_partner\_advertisements\_get, 979  
vtss\_phy\_eee\_power\_save\_state\_get, 978  
vtss\_phy\_enhanced\_led\_control\_init, 974  
vtss\_phy\_enhanced\_led\_control\_init\_get, 975  
vtss\_phy\_event\_enable\_get, 979  
vtss\_phy\_event\_enable\_set, 979  
vtss\_phy\_event\_poll, 980  
vtss\_phy\_ext\_connector\_loopback, 990  
vtss\_phy\_ext\_event\_poll, 1001  
vtss\_phy\_fast\_link\_fail\_t, 951  
vtss\_phy\_fefi\_detect, 999  
vtss\_phy\_fefi\_get, 998  
vtss\_phy\_fefi\_set, 998  
vtss\_phy\_flowcontrol\_decode\_status, 987  
vtss\_phy\_forced\_reset\_t, 950  
vtss\_phy\_freq\_t, 955  
vtss\_phy\_gpio\_get, 971  
vtss\_phy\_gpio\_mode, 971  
vtss\_phy\_gpio\_mode\_t, 956  
vtss\_phy\_gpio\_set, 972  
vtss\_phy\_i2c\_read, 966  
vtss\_phy\_i2c\_write, 967  
vtss\_phy\_id\_get, 962  
vtss\_phy\_is\_8051\_crc\_ok, 985  
vtss\_phy\_is\_viper\_revB, 1001  
vtss\_phy\_lcpll\_status\_get, 995  
vtss\_phy\_led\_intensity, 948  
vtss\_phy\_led\_intensity\_get, 974  
vtss\_phy\_led\_intensity\_set, 973  
vtss\_phy\_led\_mode\_set, 973  
vtss\_phy\_led\_mode\_t, 948  
vtss\_phy\_loopback\_get, 984  
vtss\_phy\_loopback\_set, 984  
vtss\_phy\_mac\_media\_inhibit\_odd\_start, 997  
vtss\_phy\_mac\_serdes\_pcs\_sgmii\_pre, 952  
vtss\_phy\_macsec\_csr\_sd6g\_rd, 1002  
vtss\_phy\_macsec\_csr\_sd6g\_wr, 1002  
vtss\_phy\_mdi\_t, 950  
vtss\_phy\_media\_force\_ams\_sel\_t, 954  
vtss\_phy\_media\_interface\_t, 949  
vtss\_phy\_media\_rem\_fault\_t, 954  
vtss\_phy\_mmd\_read, 968  
vtss\_phy\_mmd\_write, 969  
vtss\_phy\_mode\_t, 951  
vtss\_phy\_mse\_1000m\_get, 1000  
vtss\_phy\_mse\_100m\_get, 999  
vtss\_phy\_patch\_settings\_get, 993  
vtss\_phy\_pkt\_mode\_t, 951  
vtss\_phy\_port\_eee\_capable, 976  
vtss\_phy\_post\_reset, 958  
vtss\_phy\_power\_conf\_get, 964  
vtss\_phy\_power\_conf\_set, 964  
vtss\_phy\_power\_status\_get, 964  
vtss\_phy\_pre\_reset, 957  
vtss\_phy\_pre\_system\_reset, 958  
vtss\_phy\_read, 967  
vtss\_phy\_read\_page, 968  
vtss\_phy\_read\_tr\_addr, 1000  
vtss\_phy\_recov\_clk\_t, 948  
vtss\_phy\_reg\_decode\_status, 987  
vtss\_phy\_reset, 958  
vtss\_phy\_reset\_get, 959

vtss\_phy\_reset\_lcpll, 995  
 vtss\_phy\_sd6g\_mac\_serdes\_conf, 1003  
 vtss\_phy\_sd6g\_ob\_lev\_rd, 997  
 vtss\_phy\_sd6g\_ob\_lev\_wr, 997  
 vtss\_phy\_sd6g\_ob\_post\_rd, 996  
 vtss\_phy\_sd6g\_ob\_post\_wr, 996  
 vtss\_phy\_serdes1g\_rcpll\_status\_get, 994  
 vtss\_phy\_serdes6g\_rcpll\_status\_get, 994  
 vtss\_phy\_serdes\_fmedia\_loopback, 991  
 vtss\_phy\_serdes\_sgmii\_loopback, 990  
 vtss\_phy\_sigdet\_polarity\_t, 952  
 vtss\_phy\_statistic\_get, 983  
 vtss\_phy\_status\_1g\_get, 963  
 vtss\_phy\_status\_get, 961  
 vtss\_phy\_status\_inst\_poll, 1002  
 vtss\_phy\_unidirectional\_t, 952  
 vtss\_phy\_veriphy\_get, 973  
 vtss\_phy\_veriphy\_start, 972  
 vtss\_phy\_warm\_start\_failed\_get, 988  
 vtss\_phy\_wol\_conf\_get, 992  
 vtss\_phy\_wol\_conf\_set, 993  
 vtss\_phy\_wol\_enable, 992  
 vtss\_phy\_write, 969  
 vtss\_phy\_write\_masked, 970  
 vtss\_phy\_write\_masked\_page, 970  
 vtss\_squelch\_workaround, 980  
 vtss\_wol\_passwd\_len\_type\_t, 957  
 vtss\_phy\_atom12\_patch\_settings\_get  
     vtss\_phy\_api.h, 986  
 vtss\_phy\_cfg\_ib\_cterm  
     vtss\_phy\_api.h, 986  
 vtss\_phy\_cfg\_ob\_post0  
     vtss\_phy\_api.h, 985  
 vtss\_phy\_chip\_temp\_get  
     vtss\_phy\_api.h, 959  
 vtss\_phy\_chip\_temp\_init  
     vtss\_phy\_api.h, 960  
 vtss\_phy\_cl37\_lp\_abil\_get  
     vtss\_phy\_api.h, 961  
 vtss\_phy\_clk\_source\_t  
     vtss\_phy\_api.h, 955  
 vtss\_phy\_clk\_squelch  
     vtss\_phy\_api.h, 955  
 vtss\_phy\_clock\_conf\_get  
     vtss\_phy\_api.h, 965  
 vtss\_phy\_clock\_conf\_set  
     vtss\_phy\_api.h, 965  
 vtss\_phy\_clock\_conf\_t, 332  
     freq, 333  
     squelch, 333  
     src, 332  
 vtss\_phy\_coma\_mode\_disable  
     vtss\_phy\_api.h, 975  
 vtss\_phy\_coma\_mode\_enable  
     vtss\_phy\_api.h, 976  
 vtss\_phy\_conf\_1g\_get  
     vtss\_phy\_api.h, 962  
 vtss\_phy\_conf\_1g\_set  
     vtss\_phy\_api.h, 963  
 vtss\_phy\_conf\_1g\_t, 333  
     cfg, 334  
     master, 334  
     val, 334  
 vtss\_phy\_conf\_get  
     vtss\_phy\_api.h, 960  
 vtss\_phy\_conf\_set  
     vtss\_phy\_api.h, 961  
 vtss\_phy\_conf\_t, 334  
     aneg, 335  
     flf, 336  
     force\_ams\_sel, 337  
     forced, 335  
     mac\_if\_pcs, 336  
     mdi, 335  
     media\_if\_pcs, 336  
     mode, 335  
     sigdet, 336  
     skip\_coma, 337  
     unidir, 336  
 vtss\_phy\_csr\_rd  
     vtss\_phy\_api.h, 981  
 vtss\_phy\_csr\_wr  
     vtss\_phy\_api.h, 981  
 vtss\_phy\_daisy\_chain\_conf\_t, 337  
     spi\_daisy\_input, 338  
     spi\_daisy\_output, 338  
 vtss\_phy\_daisy\_conf\_get  
     vtss\_phy\_ts\_api.h, 1049  
 vtss\_phy\_daisy\_conf\_set  
     vtss\_phy\_ts\_api.h, 1049  
 vtss\_phy\_debug\_phyinfo\_print  
     vtss\_phy\_api.h, 989  
 vtss\_phy\_debug\_regdump\_print  
     vtss\_phy\_api.h, 991  
 vtss\_phy\_debug\_register\_dump  
     vtss\_phy\_api.h, 989  
 vtss\_phy\_debug\_stat\_print  
     vtss\_phy\_api.h, 988  
 vtss\_phy\_detect\_base\_ports  
     vtss\_phy\_api.h, 990  
 vtss\_phy\_do\_page\_chk\_get  
     vtss\_phy\_api.h, 984  
 vtss\_phy\_do\_page\_chk\_set  
     vtss\_phy\_api.h, 983  
 vtss\_phy\_eee\_conf\_get  
     vtss\_phy\_api.h, 977  
 vtss\_phy\_eee\_conf\_set  
     vtss\_phy\_api.h, 978  
 vtss\_phy\_eee\_conf\_t, 338  
     eee\_ena\_phy, 339  
     eee\_mode, 338  
 vtss\_phy\_eee\_ena  
     vtss\_phy\_api.h, 977  
 vtss\_phy\_eee\_link\_partner\_advertisements\_get  
     vtss\_phy\_api.h, 979  
 vtss\_phy\_eee\_power\_save\_state\_get

vtss\_phy\_api.h, 978  
vtss\_phy\_enhanced\_led\_control\_init  
    vtss\_phy\_api.h, 974  
vtss\_phy\_enhanced\_led\_control\_init\_get  
    vtss\_phy\_api.h, 975  
vtss\_phy\_enhanced\_led\_control\_t, 339  
    ser\_led\_frame\_rate, 340  
    ser\_led\_output\_1, 339  
    ser\_led\_output\_2, 340  
    ser\_led\_select, 340  
vtss\_phy\_event\_enable\_get  
    vtss\_phy\_api.h, 979  
vtss\_phy\_event\_enable\_set  
    vtss\_phy\_api.h, 979  
vtss\_phy\_event\_poll  
    vtss\_phy\_api.h, 980  
vtss\_phy\_ext\_connector\_loopback  
    vtss\_phy\_api.h, 990  
vtss\_phy\_ext\_event\_poll  
    vtss\_phy\_api.h, 1001  
vtss\_phy\_fast\_link\_fail\_t  
    vtss\_phy\_api.h, 951  
vtss\_phy\_fefi\_detect  
    vtss\_phy\_api.h, 999  
vtss\_phy\_fefi\_get  
    vtss\_phy\_api.h, 998  
vtss\_phy\_fefi\_set  
    vtss\_phy\_api.h, 998  
vtss\_phy\_flowcontrol\_decode\_status  
    vtss\_phy\_api.h, 987  
vtss\_phy\_forced\_reset\_t  
    vtss\_phy\_api.h, 950  
vtss\_phy\_forced\_t, 340  
    fdx, 341  
    speed, 341  
vtss\_phy\_freq\_t  
    vtss\_phy\_api.h, 955  
vtss\_phy\_gpio\_get  
    vtss\_phy\_api.h, 971  
vtss\_phy\_gpio\_mode  
    vtss\_phy\_api.h, 971  
vtss\_phy\_gpio\_mode\_t  
    vtss\_phy\_api.h, 956  
vtss\_phy\_gpio\_set  
    vtss\_phy\_api.h, 972  
vtss\_phy\_i2c\_read  
    vtss\_phy\_api.h, 966  
vtss\_phy\_i2c\_write  
    vtss\_phy\_api.h, 967  
vtss\_phy\_id\_get  
    vtss\_phy\_api.h, 962  
vtss\_phy\_interface\_mode  
    vtss\_phy\_10g\_api.h, 863  
vtss\_phy\_is\_8051\_crc\_ok  
    vtss\_phy\_api.h, 985  
vtss\_phy\_is\_viper\_revB  
    vtss\_phy\_api.h, 1001  
vtss\_phy\_lcpll\_status\_get  
    vtss\_phy\_api.h, 995  
vtss\_phy\_led\_intensity  
    vtss\_phy\_api.h, 948  
vtss\_phy\_led\_intensity\_get  
    vtss\_phy\_api.h, 974  
vtss\_phy\_led\_intensity\_set  
    vtss\_phy\_api.h, 973  
vtss\_phy\_led\_mode\_select\_t, 341  
    mode, 342  
    number, 342  
vtss\_phy\_led\_mode\_set  
    vtss\_phy\_api.h, 973  
vtss\_phy\_led\_mode\_t  
    vtss\_phy\_api.h, 948  
vtss\_phy\_loopback\_get  
    vtss\_phy\_api.h, 984  
vtss\_phy\_loopback\_set  
    vtss\_phy\_api.h, 984  
vtss\_phy\_loopback\_t, 342  
    connector\_enable, 343  
    far\_end\_enable, 343  
    mac\_serdes\_equipment\_enable, 344  
    mac\_serdes\_facility\_enable, 343  
    mac\_serdes\_input\_enable, 343  
    media\_serdes\_equipment\_enable, 344  
    media\_serdes\_facility\_enable, 344  
    media\_serdes\_input\_enable, 344  
    near\_end\_enable, 343  
vtss\_phy\_ltc\_freq\_synth\_s, 345  
    enable, 345  
    high\_duty\_cycle, 345  
    low\_duty\_cycle, 345  
vtss\_phy\_mac\_media\_inhibit\_odd\_start  
    vtss\_phy\_api.h, 997  
vtss\_phy\_mac\_serd\_pcs\_cntl\_t, 346  
    aneg\_restart, 347  
    disable, 346  
    fast\_link\_stat\_ena, 348  
    force\_adv\_ability, 347  
    inhibit\_odd\_start, 349  
    pd\_enable, 347  
    restart, 347  
    serdes\_aneg\_ena, 348  
    serdes\_pol\_inv\_in, 348  
    serdes\_pol\_inv\_out, 348  
    sgmii\_in\_pre, 347  
    sgmii\_out\_pre, 348  
vtss\_phy\_mac\_serd\_pcs\_sgmii\_pre  
    vtss\_phy\_api.h, 952  
vtss\_phy\_macsec\_csr\_sd6g\_rd  
    vtss\_phy\_api.h, 1002  
vtss\_phy\_macsec\_csr\_sd6g\_wr  
    vtss\_phy\_api.h, 1002  
vtss\_phy\_mdi\_t  
    vtss\_phy\_api.h, 950  
vtss\_phy\_media\_force\_ams\_sel\_t  
    vtss\_phy\_api.h, 954  
vtss\_phy\_media\_interface\_t

vtss\_phy\_api.h, 949  
 vtss\_phy\_media\_rem\_fault\_t  
     vtss\_phy\_api.h, 954  
 vtss\_phy\_media\_serd\_pcs\_cntl\_t, 349  
     aneg\_pd\_detect, 350  
     force\_adv\_ability, 350  
     force\_fefi, 351  
     force\_fefi\_value, 351  
     force\_hls, 351  
     inhibit\_odd\_start, 350  
     remote\_fault, 350  
     serdes\_pol\_inv\_in, 350  
     serdes\_pol\_inv\_out, 350  
 vtss\_phy\_mmd\_read  
     vtss\_phy\_api.h, 968  
 vtss\_phy\_mmd\_write  
     vtss\_phy\_api.h, 969  
 vtss\_phy\_mode\_t  
     vtss\_phy\_api.h, 951  
 vtss\_phy\_mse\_1000m\_get  
     vtss\_phy\_api.h, 1000  
 vtss\_phy\_mse\_100m\_get  
     vtss\_phy\_api.h, 999  
 vtss\_phy\_patch\_settings\_get  
     vtss\_phy\_api.h, 993  
 vtss\_phy\_pcs\_cnt\_t, 351  
     ber\_cnt, 352  
     block\_lock\_latched, 352  
     err\_blk\_cnt, 352  
     high\_ber\_latched, 352  
 vtss\_phy\_pkt\_mode\_t  
     vtss\_phy\_api.h, 951  
 vtss\_phy\_port\_eee\_capable  
     vtss\_phy\_api.h, 976  
 vtss\_phy\_post\_reset  
     vtss\_phy\_api.h, 958  
 vtss\_phy\_power\_conf\_get  
     vtss\_phy\_api.h, 964  
 vtss\_phy\_power\_conf\_set  
     vtss\_phy\_api.h, 964  
 vtss\_phy\_power\_conf\_t, 353  
     mode, 353  
 vtss\_phy\_power\_mode\_t  
     phy.h, 579  
 vtss\_phy\_power\_status\_get  
     vtss\_phy\_api.h, 964  
 vtss\_phy\_power\_status\_t, 354  
     level, 354  
 vtss\_phy\_pre\_reset  
     vtss\_phy\_api.h, 957  
 vtss\_phy\_pre\_system\_reset  
     vtss\_phy\_api.h, 958  
 vtss\_phy\_read  
     vtss\_phy\_api.h, 967  
 vtss\_phy\_read\_page  
     vtss\_phy\_api.h, 968  
 vtss\_phy\_read\_tr\_addr  
     vtss\_phy\_api.h, 1000  
 vtss\_phy\_recov\_clk\_t  
     vtss\_phy\_api.h, 948  
 vtss\_phy\_reg\_decode\_status  
     vtss\_phy\_api.h, 987  
 vtss\_phy\_reset  
     vtss\_phy\_api.h, 958  
 vtss\_phy\_reset\_conf\_t, 354  
     force, 355  
     i\_cpu\_en, 356  
     mac\_if, 355  
     media\_if, 355  
     pkt\_mode, 356  
     rgmii, 355  
     tbi, 355  
 vtss\_phy\_reset\_get  
     vtss\_phy\_api.h, 959  
 vtss\_phy\_reset\_lcpll  
     vtss\_phy\_api.h, 995  
 vtss\_phy\_rgmii\_conf\_t, 356  
     rx\_clk\_skew\_ps, 357  
     tx\_clk\_skew\_ps, 357  
 vtss\_phy\_sd6g\_mac\_serdes\_conf  
     vtss\_phy\_api.h, 1003  
 vtss\_phy\_sd6g\_ob\_lev\_rd  
     vtss\_phy\_api.h, 997  
 vtss\_phy\_sd6g\_ob\_lev\_wr  
     vtss\_phy\_api.h, 997  
 vtss\_phy\_sd6g\_ob\_post\_rd  
     vtss\_phy\_api.h, 996  
 vtss\_phy\_sd6g\_ob\_post\_wr  
     vtss\_phy\_api.h, 996  
 vtss\_phy\_serdes1g\_rcpll\_status\_get  
     vtss\_phy\_api.h, 994  
 vtss\_phy\_serdes6g\_rcpll\_status\_get  
     vtss\_phy\_api.h, 994  
 vtss\_phy\_serdes\_fmedia\_loopback  
     vtss\_phy\_api.h, 991  
 vtss\_phy\_serdes\_sgmii\_loopback  
     vtss\_phy\_api.h, 990  
 vtss\_phy\_sigdet\_polarity\_t  
     vtss\_phy\_api.h, 952  
 vtss\_phy\_statistic\_get  
     vtss\_phy\_api.h, 983  
 vtss\_phy\_statistic\_t, 357  
     cu\_bad, 358  
     cu\_good, 358  
     media\_mac\_serdes\_crc, 359  
     media\_mac\_serdes\_good, 359  
     rx\_err\_cnt\_base\_tx, 358  
     serdes\_tx\_bad, 358  
     serdes\_tx\_good, 358  
 vtss\_phy\_status\_1g\_get  
     vtss\_phy\_api.h, 963  
 vtss\_phy\_status\_1g\_t, 359  
     master, 360  
     master\_cfg\_fault, 360  
 vtss\_phy\_status\_get  
     vtss\_phy\_api.h, 961

vtss\_phy\_status\_inst\_poll  
  vtss\_phy\_api.h, 1002

vtss\_phy\_tbi\_conf\_t, 360  
  aneg\_enable, 361

vtss\_phy\_timestamp\_t, 361  
  high, 361  
  low, 361  
  nanoseconds, 362  
  seconds, 362

vtss\_phy\_ts\_10g\_extended\_fifo\_sync  
  vtss\_phy\_ts\_api.h, 1071

vtss\_phy\_ts\_10g\_fifo\_sync  
  vtss\_phy\_ts\_api.h, 1071

vtss\_phy\_ts\_8487\_xaui\_sel\_t  
  vtss\_phy\_ts\_api.h, 1026

vtss\_phy\_ts\_ach\_conf\_t, 362  
  channel\_type, 364  
  comm\_opt, 364  
  mask, 363  
  proto\_id, 364  
  value, 363  
  version, 363

vtss\_phy\_ts\_action\_format  
  vtss\_phy\_ts\_api.h, 1031

vtss\_phy\_ts\_alt\_clock\_mode\_get  
  vtss\_phy\_ts\_api.h, 1035

vtss\_phy\_ts\_alt\_clock\_mode\_s, 364  
  ls\_lpbk, 365  
  ls\_pps\_lpbk, 365  
  pps\_ls\_lpbk, 365

vtss\_phy\_ts\_alt\_clock\_mode\_set  
  vtss\_phy\_ts\_api.h, 1036

vtss\_phy\_ts\_alt\_clock\_mode\_t  
  vtss\_phy\_ts\_api.h, 1025

vtss\_phy\_ts\_alt\_clock\_saved\_get  
  vtss\_phy\_ts\_api.h, 1035

vtss\_phy\_ts\_api.h  
  VTSS\_PHY\_TS\_8487\_XAUI\_SEL\_0, 1023  
  VTSS\_PHY\_TS\_8487\_XAUI\_SEL\_1, 1024  
  VTSS\_PHY\_TS\_DATA\_IN\_RSRVD\_FIELD, 1023  
  VTSS\_PHY\_TS\_EGR\_DATAPATH\_RESET, 1024  
  VTSS\_PHY\_TS\_EGR\_ENGINE\_ERR, 1022  
  VTSS\_PHY\_TS\_EGR\_FIFO\_OVERFLOW, 1023  
  VTSS\_PHY\_TS\_EGR\_FIFO\_RESET, 1025  
  VTSS\_PHY\_TS\_EGR\_LTC2\_RESET, 1024  
  VTSS\_PHY\_TS\_EGR\_RW\_FCS\_ERR, 1022  
  VTSS\_PHY\_TS\_EGR\_TIMESTAMP\_CAPTURED, 1022  
  VTSS\_PHY\_TS\_ENG\_FLOW\_VALID\_FOR\_CH0, 1021  
  VTSS\_PHY\_TS\_ENG\_FLOW\_VALID\_FOR\_CH1, 1021  
  VTSS\_PHY\_TS\_ETH\_ADDR\_MATCH\_48BIT, 1016  
  VTSS\_PHY\_TS\_ETH\_ADDR\_MATCH\_ANY\_M←  
    ULTICAST, 1017  
  VTSS\_PHY\_TS\_ETH\_ADDR\_MATCH\_ANY\_U←  
    NICAST, 1016

VTSS\_PHY\_TS\_ETH\_MATCH\_DEST\_ADDR,  
  1017

VTSS\_PHY\_TS\_ETH\_MATCH\_SRC\_ADDR, 1017

VTSS\_PHY\_TS\_ETH\_MATCH\_SRC\_OR\_DEST,  
  1017

VTSS\_PHY\_TS\_FIFO\_SIG\_DEST\_IP, 1013

VTSS\_PHY\_TS\_FIFO\_SIG\_DEST\_MAC, 1014

VTSS\_PHY\_TS\_FIFO\_SIG\_DOMAIN\_NUM, 1013

VTSS\_PHY\_TS\_FIFO\_SIG\_MSG\_TYPE, 1013

VTSS\_PHY\_TS\_FIFO\_SIG\_SEQ\_ID, 1014

VTSS\_PHY\_TS\_FIFO\_SIG\_SOURCE\_PORT\_ID,  
  1013

VTSS\_PHY\_TS\_FIFO\_SIG\_SRC\_IP, 1013

VTSS\_PHY\_TS\_INGR\_DATAPATH\_RESET,  
  1024

VTSS\_PHY\_TS\_INGR\_ENGINE\_ERR, 1021

VTSS\_PHY\_TS\_INGR\_LTC1\_RESET, 1024

VTSS\_PHY\_TS\_INGR\_RW\_FCS\_ERR, 1022

VTSS\_PHY\_TS\_INGR\_RW\_PREAM\_ERR, 1022

VTSS\_PHY\_TS\_IP\_MATCH\_DEST, 1019

VTSS\_PHY\_TS\_IP\_MATCH\_SRC\_OR\_DEST,  
  1020

VTSS\_PHY\_TS\_IP\_MATCH\_SRC, 1019

VTSS\_PHY\_TS\_IP\_VER\_4, 1019

VTSS\_PHY\_TS\_IP\_VER\_6, 1019

VTSS\_PHY\_TS\_LTC\_LOAD\_SAVE\_NEW\_TOD,  
  1023

VTSS\_PHY\_TS\_LTC\_NEW\_PPS\_INTRPT, 1023

VTSS\_PHY\_TS MPLS\_STACK\_DEPTH\_1, 1020

VTSS\_PHY\_TS MPLS\_STACK\_DEPTH\_2, 1020

VTSS\_PHY\_TS MPLS\_STACK\_DEPTH\_3, 1020

VTSS\_PHY\_TS MPLS\_STACK\_DEPTH\_4, 1020

VTSS\_PHY\_TS MPLS\_STACK\_REF\_POINT←  
  END, 1021

VTSS\_PHY\_TS MPLS\_STACK\_REF\_POINT←  
  TOP, 1021

VTSS\_PHY\_TS SIG\_DEST\_IP\_LEN, 1015

VTSS\_PHY\_TS SIG\_DEST\_MAC\_LEN, 1016

VTSS\_PHY\_TS SIG\_DOMAIN\_NUM\_LEN, 1014

VTSS\_PHY\_TS SIG\_LEN, 1014

VTSS\_PHY\_TS SIG\_MSG\_TYPE\_LEN, 1015

VTSS\_PHY\_TS SIG\_SEQ\_ID\_LEN, 1015

VTSS\_PHY\_TS SIG\_SEQUENCE\_ID\_LEN, 1015

VTSS\_PHY\_TS SIG\_SOURCE\_PORT\_ID\_LEN,  
  1015

VTSS\_PHY\_TS SIG\_SRC\_IP\_LEN, 1016

VTSS\_PHY\_TS SIG\_TIME\_STAMP\_LEN, 1014

VTSS\_PHY\_TS TAG\_RANGE\_INNER, 1019

VTSS\_PHY\_TS TAG\_RANGE\_NONE, 1018

VTSS\_PHY\_TS TAG\_RANGE\_OUTER, 1018

VTSS\_PHY\_TS TAG\_TYPE\_B, 1018

VTSS\_PHY\_TS TAG\_TYPE\_C, 1017

VTSS\_PHY\_TS TAG\_TYPE\_I, 1018

VTSS\_PHY\_TS TAG\_TYPE\_S, 1018

VTSS\_PTP\_IP\_1588\_VERSION\_2\_1, 1016

vtss\_phy\_1588\_csr\_reg\_read, 1070

vtss\_phy\_1588\_csr\_reg\_write, 1069

vtss\_phy\_1588\_debug\_reg\_read, 1073

vtss\_phy\_1g\_ts\_fifo\_sync, 1073  
 vtss\_phy\_daisy\_conf\_get, 1049  
 vtss\_phy\_daisy\_conf\_set, 1049  
 vtss\_phy\_ts\_10g\_extended\_fifo\_sync, 1071  
 vtss\_phy\_ts\_10g\_fifo\_sync, 1071  
 vtss\_phy\_ts\_8487\_xaui\_sel\_t, 1026  
 vtss\_phy\_ts\_action\_format, 1031  
 vtss\_phy\_ts\_alt\_clock\_mode\_get, 1035  
 vtss\_phy\_ts\_alt\_clock\_mode\_set, 1036  
 vtss\_phy\_ts\_alt\_clock\_mode\_t, 1025  
 vtss\_phy\_ts\_alt\_clock\_saved\_get, 1035  
 vtss\_phy\_ts\_block\_soft\_reset, 1067  
 vtss\_phy\_ts\_bypass\_clear, 1071  
 vtss\_phy\_ts\_clock\_rateadj\_get, 1045  
 vtss\_phy\_ts\_clock\_rateadj\_ppm\_get, 1046  
 vtss\_phy\_ts\_clock\_rateadj\_ppm\_set, 1046  
 vtss\_phy\_ts\_clock\_rateadj\_set, 1045  
 vtss\_phy\_ts\_clock\_src\_t, 1032  
 vtss\_phy\_ts\_clockfreq\_t, 1031  
 vtss\_phy\_ts\_correction\_overflow\_get, 1064  
 vtss\_phy\_ts\_delay\_asymmetry\_get, 1041  
 vtss\_phy\_ts\_delay\_asymmetry\_set, 1040  
 vtss\_phy\_ts\_egress\_engine\_action\_get, 1060  
 vtss\_phy\_ts\_egress\_engine\_action\_set, 1060  
 vtss\_phy\_ts\_egress\_engine\_clear, 1055  
 vtss\_phy\_ts\_egress\_engine\_conf\_get, 1057  
 vtss\_phy\_ts\_egress\_engine\_conf\_set, 1057  
 vtss\_phy\_ts\_egress\_engine\_init, 1054  
 vtss\_phy\_ts\_egress\_engine\_init\_conf\_get, 1055  
 vtss\_phy\_ts\_egress\_latency\_get, 1039  
 vtss\_phy\_ts\_egress\_latency\_set, 1038  
 vtss\_phy\_ts\_encap\_t, 1027  
 vtss\_phy\_ts\_engine\_channel\_map\_t, 1026  
 vtss\_phy\_ts\_engine\_flow\_match\_t, 1028  
 vtss\_phy\_ts\_engine\_t, 1028  
 vtss\_phy\_ts\_event\_enable\_get, 1062  
 vtss\_phy\_ts\_event\_enable\_set, 1062  
 vtss\_phy\_ts\_event\_poll, 1063  
 vtss\_phy\_ts\_event\_t, 1026  
 vtss\_phy\_ts\_fifo\_empty, 1051  
 vtss\_phy\_ts\_fifo\_mode\_t, 1033  
 vtss\_phy\_ts\_fifo\_read, 1026  
 vtss\_phy\_ts\_fifo\_read\_cb\_get, 1052  
 vtss\_phy\_ts\_fifo\_read\_install, 1051  
 vtss\_phy\_ts\_fifo\_sig\_get, 1050  
 vtss\_phy\_ts\_fifo\_sig\_mask\_t, 1025  
 vtss\_phy\_ts\_fifo\_sig\_set, 1050  
 vtss\_phy\_ts\_fifo\_status\_t, 1027  
 vtss\_phy\_ts\_fifo\_timestamp\_len\_t, 1033  
 vtss\_phy\_ts\_flow\_clear\_cf\_set, 1074  
 vtss\_phy\_ts\_hiacc\_get, 1067  
 vtss\_phy\_ts\_hiacc\_set, 1066  
 vtss\_phy\_ts\_ietf\_mpls\_ach\_oam\_delaym\_type\_t, 1030  
 vtss\_phy\_ts\_ietf\_mpls\_ach\_oam\_ts\_format\_← t, 1031  
 vtss\_phy\_ts\_ingress\_engine\_action\_get, 1058  
 vtss\_phy\_ts\_ingress\_engine\_action\_set, 1058  
 vtss\_phy\_ts\_ingress\_engine\_clear, 1053  
 vtss\_phy\_ts\_ingress\_engine\_conf\_get, 1056  
 vtss\_phy\_ts\_ingress\_engine\_conf\_set, 1055  
 vtss\_phy\_ts\_ingress\_engine\_init, 1052  
 vtss\_phy\_ts\_ingress\_engine\_init\_conf\_get, 1053  
 vtss\_phy\_ts\_ingress\_latency\_get, 1038  
 vtss\_phy\_ts\_ingress\_latency\_set, 1037  
 vtss\_phy\_ts\_init, 1065  
 vtss\_phy\_ts\_init\_conf\_get, 1065  
 vtss\_phy\_ts\_load\_ptptime\_get, 1043  
 vtss\_phy\_ts\_loadpulse\_delay\_get, 1045  
 vtss\_phy\_ts\_loadpulse\_delay\_set, 1044  
 vtss\_phy\_ts\_ltc\_freq\_synth\_pulse\_get, 1048  
 vtss\_phy\_ts\_ltc\_freq\_synth\_pulse\_set, 1048  
 vtss\_phy\_ts\_mode\_get, 1064  
 vtss\_phy\_ts\_mode\_set, 1064  
 vtss\_phy\_ts\_new\_spi\_mode\_get, 1068  
 vtss\_phy\_ts\_new\_spi\_mode\_set, 1068  
 vtss\_phy\_ts\_nphase\_sampler\_t, 1034  
 vtss\_phy\_ts\_nphase\_status\_get, 1066  
 vtss\_phy\_ts\_ongoing\_adjustment, 1048  
 vtss\_phy\_ts\_path\_delay\_get, 1040  
 vtss\_phy\_ts\_path\_delay\_set, 1039  
 vtss\_phy\_ts\_phy\_oper\_mode\_change, 1069  
 vtss\_phy\_ts\_pps\_conf\_get, 1037  
 vtss\_phy\_ts\_pps\_conf\_set, 1036  
 vtss\_phy\_ts\_ptp\_clock\_mode\_t, 1029  
 vtss\_phy\_ts\_ptp\_delaym\_type\_t, 1029  
 vtss\_phy\_ts\_ptp\_message\_type\_t, 1034  
 vtss\_phy\_ts\_ptptime\_adj1ns, 1047  
 vtss\_phy\_ts\_ptptime\_arm, 1042  
 vtss\_phy\_ts\_ptptime\_get, 1042  
 vtss\_phy\_ts\_ptptime\_set, 1041  
 vtss\_phy\_ts\_ptptime\_set\_done, 1042  
 vtss\_phy\_ts\_rxtimestamp\_len\_t, 1032  
 vtss\_phy\_ts\_rxtimestamp\_pos\_t, 1032  
 vtss\_phy\_ts\_scaled\_ppb\_t, 1025  
 vtss\_phy\_ts\_sertod\_get, 1044  
 vtss\_phy\_ts\_sertod\_set, 1043  
 vtss\_phy\_ts\_soft\_reset\_t, 1026  
 vtss\_phy\_ts\_stats\_get, 1063  
 vtss\_phy\_ts\_status\_check, 1070  
 vtss\_phy\_ts\_tc\_op\_mode\_t, 1034  
 vtss\_phy\_ts\_tesla\_tsp\_fifo\_sync, 1072  
 vtss\_phy\_ts\_timeofday\_offset\_set, 1047  
 vtss\_phy\_ts\_todadj\_status\_t, 1027  
 vtss\_phy\_ts\_viper\_fifo\_reset, 1072  
 vtss\_phy\_ts\_y1731\_oam\_delaym\_type\_t, 1030  
 vtss\_phy\_ts\_block\_soft\_reset  
     vtss\_phy\_ts\_api.h, 1067  
 vtss\_phy\_ts\_bypass\_clear  
     vtss\_phy\_ts\_api.h, 1071  
 vtss\_phy\_ts\_clock\_rateadj\_get  
     vtss\_phy\_ts\_api.h, 1045  
 vtss\_phy\_ts\_clock\_rateadj\_ppm\_get  
     vtss\_phy\_ts\_api.h, 1046  
 vtss\_phy\_ts\_clock\_rateadj\_ppm\_set  
     vtss\_phy\_ts\_api.h, 1046

vtss\_phy\_ts\_clock\_rateadj\_set  
    vtss\_phy\_ts\_api.h, 1045  
vtss\_phy\_ts\_clock\_src\_t  
    vtss\_phy\_ts\_api.h, 1032  
vtss\_phy\_ts\_clockfreq\_t  
    vtss\_phy\_ts\_api.h, 1031  
vtss\_phy\_ts\_correction\_overflow\_get  
    vtss\_phy\_ts\_api.h, 1064  
vtss\_phy\_ts\_delay\_asymmetry\_get  
    vtss\_phy\_ts\_api.h, 1041  
vtss\_phy\_ts\_delay\_asymmetry\_set  
    vtss\_phy\_ts\_api.h, 1040  
vtss\_phy\_ts\_egress\_engine\_action\_get  
    vtss\_phy\_ts\_api.h, 1060  
vtss\_phy\_ts\_egress\_engine\_action\_set  
    vtss\_phy\_ts\_api.h, 1060  
vtss\_phy\_ts\_egress\_engine\_clear  
    vtss\_phy\_ts\_api.h, 1055  
vtss\_phy\_ts\_egress\_engine\_conf\_get  
    vtss\_phy\_ts\_api.h, 1057  
vtss\_phy\_ts\_egress\_engine\_conf\_set  
    vtss\_phy\_ts\_api.h, 1057  
vtss\_phy\_ts\_egress\_engine\_init  
    vtss\_phy\_ts\_api.h, 1054  
vtss\_phy\_ts\_egress\_engine\_init\_conf\_get  
    vtss\_phy\_ts\_api.h, 1055  
vtss\_phy\_ts\_egress\_latency\_get  
    vtss\_phy\_ts\_api.h, 1039  
vtss\_phy\_ts\_egress\_latency\_set  
    vtss\_phy\_ts\_api.h, 1038  
vtss\_phy\_ts\_encap\_t  
    vtss\_phy\_ts\_api.h, 1027  
vtss\_phy\_ts\_eng\_init\_conf\_t, 365  
    encap\_type, 366  
    eng\_used, 366  
    flow\_end\_index, 367  
    flow\_match\_mode, 366  
    flow\_st\_index, 366  
vtss\_phy\_ts\_engine\_action\_t, 367  
    action, 368  
    action\_gen, 368  
    action\_ptp, 368  
    gen\_conf, 368  
    oam\_conf, 368  
    ptp\_conf, 368  
vtss\_phy\_ts\_engine\_channel\_map\_t  
    vtss\_phy\_ts\_api.h, 1026  
vtss\_phy\_ts\_engine\_flow\_conf\_t, 369  
    channel\_map, 369  
    eng\_mode, 369  
    flow\_conf, 370  
    gen, 370  
    oam, 370  
    ptp, 370  
vtss\_phy\_ts\_engine\_flow\_match\_t  
    vtss\_phy\_ts\_api.h, 1028  
vtss\_phy\_ts\_engine\_t  
    vtss\_phy\_ts\_api.h, 1028

vtss\_phy\_ts\_eth\_conf\_t, 371  
    addr\_match\_mode, 373  
    addr\_match\_select, 373  
    comm\_opt, 372  
    etype, 372  
    flow\_en, 372  
    flow\_opt, 376  
    i\_tag, 376  
    inner\_tag, 376  
    inner\_tag\_type, 374  
    lower, 374  
    mac\_addr, 373  
    mask, 375, 376  
    num\_tag, 373  
    outer\_tag, 375  
    outer\_tag\_type, 374  
    pbb\_en, 372  
    range, 375  
    tag\_range\_mode, 374  
    tpid, 372  
    upper, 374  
    val, 375, 376  
    value, 375, 376  
    vlan\_check, 373

vtss\_phy\_ts\_event\_enable\_get  
    vtss\_phy\_ts\_api.h, 1062  
vtss\_phy\_ts\_event\_enable\_set  
    vtss\_phy\_ts\_api.h, 1062  
vtss\_phy\_ts\_event\_poll  
    vtss\_phy\_ts\_api.h, 1063  
vtss\_phy\_ts\_event\_t  
    vtss\_phy\_ts\_api.h, 1026  
vtss\_phy\_ts\_fifo\_conf\_t, 377  
    detect\_only, 377  
    eng\_minE, 378  
    eng\_recov, 377  
    skip\_rev\_check, 378  
vtss\_phy\_ts\_fifo\_empty  
    vtss\_phy\_ts\_api.h, 1051  
vtss\_phy\_ts\_fifo\_mode\_t  
    vtss\_phy\_ts\_api.h, 1033  
vtss\_phy\_ts\_fifo\_read  
    vtss\_phy\_ts\_api.h, 1026  
vtss\_phy\_ts\_fifo\_read\_cb\_get  
    vtss\_phy\_ts\_api.h, 1052  
vtss\_phy\_ts\_fifo\_read\_install  
    vtss\_phy\_ts\_api.h, 1051  
vtss\_phy\_ts\_fifo\_sig\_get  
    vtss\_phy\_ts\_api.h, 1050  
vtss\_phy\_ts\_fifo\_sig\_mask\_t  
    vtss\_phy\_ts\_api.h, 1025  
vtss\_phy\_ts\_fifo\_sig\_set  
    vtss\_phy\_ts\_api.h, 1050  
vtss\_phy\_ts\_fifo\_sig\_t, 378  
    dest\_ip, 380  
    dest\_mac, 380  
    domain\_num, 379  
    msg\_type, 379

sequence\_id, 379  
 sig\_mask, 379  
 src\_ip, 380  
 src\_port\_identity, 379  
 vtss\_phy\_ts\_fifo\_status\_t  
     vtss\_phy\_ts\_api.h, 1027  
 vtss\_phy\_ts\_fifo\_timestamp\_len\_t  
     vtss\_phy\_ts\_api.h, 1033  
 vtss\_phy\_ts\_flow\_clear\_cf\_set  
     vtss\_phy\_ts\_api.h, 1074  
 vtss\_phy\_ts\_gen\_conf\_t, 380  
     comm\_opt, 381  
     data, 382  
     flow\_en, 381  
     flow\_offset, 381  
     flow\_opt, 382  
     mask, 382  
     next\_prot\_offset, 381  
 vtss\_phy\_ts\_generic\_action\_t, 382  
     channel\_map, 383  
     data, 383  
     enable, 383  
     flow\_id, 383  
     mask, 384  
     ts\_offset, 384  
     ts\_type, 384  
 vtss\_phy\_ts\_generic\_flow\_conf\_t, 384  
     eth1\_opt, 385  
     gen\_opt, 385  
 vtss\_phy\_ts\_hiacc\_get  
     vtss\_phy\_ts\_api.h, 1067  
 vtss\_phy\_ts\_hiacc\_set  
     vtss\_phy\_ts\_api.h, 1066  
 vtss\_phy\_ts\_ietf\_mpls\_ach\_oam\_conf\_t, 385  
     delaym\_type, 386  
     ds, 386  
     ts\_format, 386  
 vtss\_phy\_ts\_ietf\_mpls\_ach\_oam\_delaym\_type\_t  
     vtss\_phy\_ts\_api.h, 1030  
 vtss\_phy\_ts\_ietf\_mpls\_ach\_oam\_ts\_format\_t  
     vtss\_phy\_ts\_api.h, 1031  
 vtss\_phy\_ts\_ingress\_engine\_action\_get  
     vtss\_phy\_ts\_api.h, 1058  
 vtss\_phy\_ts\_ingress\_engine\_action\_set  
     vtss\_phy\_ts\_api.h, 1058  
 vtss\_phy\_ts\_ingress\_engine\_clear  
     vtss\_phy\_ts\_api.h, 1053  
 vtss\_phy\_ts\_ingress\_engine\_conf\_get  
     vtss\_phy\_ts\_api.h, 1056  
 vtss\_phy\_ts\_ingress\_engine\_conf\_set  
     vtss\_phy\_ts\_api.h, 1055  
 vtss\_phy\_ts\_ingress\_engine\_init  
     vtss\_phy\_ts\_api.h, 1052  
 vtss\_phy\_ts\_ingress\_engine\_init\_conf\_get  
     vtss\_phy\_ts\_api.h, 1053  
 vtss\_phy\_ts\_ingress\_latency\_get  
     vtss\_phy\_ts\_api.h, 1038  
 vtss\_phy\_ts\_ingress\_latency\_set  
     vtss\_phy\_ts\_api.h, 1033  
     vtss\_phy\_ts\_init  
         vtss\_phy\_ts\_api.h, 1065  
 vtss\_phy\_ts\_init\_conf\_get  
     vtss\_phy\_ts\_api.h, 1065  
 vtss\_phy\_ts\_init\_conf\_t, 386  
     auto\_clear\_ls, 389  
     chk\_ing\_modified, 390  
     clk\_freq, 387  
     clk\_src, 387  
     macsec\_ena, 389  
     one\_step\_txfifo, 390  
     rx\_ts\_len, 388  
     rx\_ts\_pos, 387  
     tc\_op\_mode, 389  
     tx\_fifo\_hi\_clk\_cycs, 388  
     tx\_fifo\_lo\_clk\_cycs, 389  
     tx\_fifo\_mode, 388  
     tx\_fifo\_spi\_conf, 388  
     tx\_ts\_len, 388  
     xaui\_sel\_8487, 389  
 vtss\_phy\_ts\_ip\_conf\_t, 390  
     addr, 393  
     comm\_opt, 392  
     dport\_mask, 392  
     dport\_val, 392  
     flow\_en, 392  
     flow\_opt, 394  
     ip\_addr, 394  
     ip\_mode, 391  
     ipv4, 393  
     ipv6, 393  
     mask, 393  
     match\_mode, 393  
     sport\_mask, 392  
     sport\_val, 391  
 vtss\_phy\_ts\_load\_ptptime\_get  
     vtss\_phy\_ts\_api.h, 1043  
 vtss\_phy\_ts\_loadpulse\_delay\_get  
     vtss\_phy\_ts\_api.h, 1045  
 vtss\_phy\_ts\_loadpulse\_delay\_set  
     vtss\_phy\_ts\_api.h, 1044  
 vtss\_phy\_ts\_ltc\_freq\_synth\_pulse\_get  
     vtss\_phy\_ts\_api.h, 1048  
 vtss\_phy\_ts\_ltc\_freq\_synth\_pulse\_set  
     vtss\_phy\_ts\_api.h, 1048  
 vtss\_phy\_ts\_mode\_get  
     vtss\_phy\_ts\_api.h, 1064  
 vtss\_phy\_ts\_mode\_set  
     vtss\_phy\_ts\_api.h, 1064  
 vtss\_phy\_ts\_mpls\_conf\_t, 394  
     bottom\_up, 397  
     comm\_opt, 395  
     cw\_en, 395  
     end, 397  
     flow\_en, 395  
     flow\_opt, 398  
     frst\_lvl\_after\_top, 396

frst\_lvl\_before\_end, 397  
snd\_lvl\_after\_top, 396  
snd\_lvl\_before\_end, 397  
stack\_depth, 395  
stack\_level, 397  
stack\_ref\_point, 395  
thrd\_lvl\_after\_top, 396  
thrd\_lvl\_before\_end, 397  
top, 396  
top\_down, 396  
vtss\_phy\_ts\_mpls\_lvl\_rng\_t, 398  
lower, 398  
upper, 398  
vtss\_phy\_ts\_new\_spi\_mode\_get  
vtss\_phy\_ts\_api.h, 1068  
vtss\_phy\_ts\_new\_spi\_mode\_set  
vtss\_phy\_ts\_api.h, 1068  
vtss\_phy\_ts\_nphase\_sampler\_t  
vtss\_phy\_ts\_api.h, 1034  
vtss\_phy\_ts\_nphase\_status\_get  
vtss\_phy\_ts\_api.h, 1066  
vtss\_phy\_ts\_nphase\_status\_t, 399  
CALIB\_DONE, 400  
CALIB\_ERR, 399  
enable, 399  
vtss\_phy\_ts\_oam\_engine\_action\_t, 400  
channel\_map, 401  
enable, 401  
ietf\_oam\_conf, 401  
oam\_conf, 402  
version, 401  
y1731\_en, 401  
y1731\_oam\_conf, 401  
vtss\_phy\_ts\_oam\_engine\_flow\_conf\_t, 402  
ach\_opt, 403  
eth1\_opt, 402  
eth2\_opt, 403  
mpls\_opt, 403  
vtss\_phy\_ts\_ongoing\_adjustment  
vtss\_phy\_ts\_api.h, 1048  
vtss\_phy\_ts\_path\_delay\_get  
vtss\_phy\_ts\_api.h, 1040  
vtss\_phy\_ts\_path\_delay\_set  
vtss\_phy\_ts\_api.h, 1039  
vtss\_phy\_ts\_phy\_oper\_mode\_change  
vtss\_phy\_ts\_api.h, 1069  
vtss\_phy\_ts\_pps\_conf\_get  
vtss\_phy\_ts\_api.h, 1037  
vtss\_phy\_ts\_pps\_conf\_set  
vtss\_phy\_ts\_api.h, 1036  
vtss\_phy\_ts\_pps\_config\_s, 403  
pps\_offset, 404  
pps\_output\_enable, 404  
pps\_width\_adj, 404  
vtss\_phy\_ts\_ptp\_clock\_mode\_t  
vtss\_phy\_ts\_api.h, 1029  
vtss\_phy\_ts\_ptp\_conf\_t, 405  
domain, 406  
lower, 406  
mask, 405  
range, 406  
range\_en, 405  
upper, 406  
val, 405  
value, 406  
vtss\_phy\_ts\_ptp\_delaym\_type\_t  
vtss\_phy\_ts\_api.h, 1029  
vtss\_phy\_ts\_ptp\_engine\_action\_t, 407  
cf\_update, 408  
channel\_map, 407  
clk\_mode, 408  
delaym\_type, 408  
enable, 407  
ptp\_conf, 407  
vtss\_phy\_ts\_ptp\_engine\_flow\_conf\_t, 408  
ach\_opt, 410  
eth1\_opt, 409  
eth2\_opt, 409  
ip1\_opt, 409  
ip2\_opt, 409  
mpls\_opt, 410  
vtss\_phy\_ts\_ptp\_message\_type\_t  
vtss\_phy\_ts\_api.h, 1034  
vtss\_phy\_ts\_ptptime\_adj1ns  
vtss\_phy\_ts\_api.h, 1047  
vtss\_phy\_ts\_ptptime\_arm  
vtss\_phy\_ts\_api.h, 1042  
vtss\_phy\_ts\_ptptime\_get  
vtss\_phy\_ts\_api.h, 1042  
vtss\_phy\_ts\_ptptime\_set  
vtss\_phy\_ts\_api.h, 1041  
vtss\_phy\_ts\_ptptime\_set\_done  
vtss\_phy\_ts\_api.h, 1042  
vtss\_phy\_ts\_rxtimestamp\_len\_t  
vtss\_phy\_ts\_api.h, 1032  
vtss\_phy\_ts\_rxtimestamp\_pos\_t  
vtss\_phy\_ts\_api.h, 1032  
vtss\_phy\_ts\_scaled\_ppb\_t  
vtss\_phy\_ts\_api.h, 1025  
vtss\_phy\_ts\_sertod\_conf\_t, 410  
ip\_enable, 411  
ls\_inv, 411  
op\_enable, 411  
vtss\_phy\_ts\_sertod\_get  
vtss\_phy\_ts\_api.h, 1044  
vtss\_phy\_ts\_sertod\_set  
vtss\_phy\_ts\_api.h, 1043  
vtss\_phy\_ts\_soft\_reset\_t  
vtss\_phy\_ts\_api.h, 1026  
vtss\_phy\_ts\_stats\_get  
vtss\_phy\_ts\_api.h, 1063  
vtss\_phy\_ts\_stats\_t, 411  
egr\_fcs\_err, 412  
egr\_frm\_mod\_cnt, 413  
egr\_pream\_shrink\_err, 412  
ingr\_fcs\_err, 412

ingr\_frm\_mod\_cnt, 413  
 ingr\_pream\_shrink\_err, 412  
 ts\_fifo\_drop\_cnt, 413  
 ts\_fifo\_tx\_cnt, 413  
 vtss\_phy\_ts\_status\_check  
     vtss\_phy\_ts\_api.h, 1070  
 vtss\_phy\_ts\_tc\_op\_mode\_t  
     vtss\_phy\_ts\_api.h, 1034  
 vtss\_phy\_ts\_tesla\_tsp\_fifo\_sync  
     vtss\_phy\_ts\_api.h, 1072  
 vtss\_phy\_ts\_timeofday\_offset\_set  
     vtss\_phy\_ts\_api.h, 1047  
 vtss\_phy\_ts\_todadj\_status\_t  
     vtss\_phy\_ts\_api.h, 1027  
 vtss\_phy\_ts\_viper\_fifo\_reset  
     vtss\_phy\_ts\_api.h, 1072  
 vtss\_phy\_ts\_y1731\_oam\_conf\_t, 414  
     delaym\_type, 414  
     lower, 415  
     mask, 415  
     meg\_level, 415  
     range, 415  
     range\_en, 414  
     upper, 415  
     val, 414  
     value, 415  
 vtss\_phy\_ts\_y1731\_oam\_delaym\_type\_t  
     vtss\_phy\_ts\_api.h, 1030  
 vtss\_phy\_type\_t, 416  
     base\_port\_no, 417  
     channel\_id, 417  
     part\_number, 416  
     phy\_api\_base\_no, 417  
     port\_cnt, 417  
     revision, 416  
 vtss\_phy\_unidirectional\_t  
     vtss\_phy\_api.h, 952  
 vtss\_phy\_veriphy\_get  
     vtss\_phy\_api.h, 973  
 vtss\_phy\_veriphy\_result\_t, 418  
     length, 418  
     link, 418  
     status, 418  
 vtss\_phy\_veriphy\_start  
     vtss\_phy\_api.h, 972  
 vtss\_phy\_veriphy\_status\_t  
     phy.h, 579  
 vtss\_phy\_warm\_start\_10g\_failed\_get  
     vtss\_phy\_10g\_api.h, 923  
 vtss\_phy\_warm\_start\_failed\_get  
     vtss\_phy\_api.h, 988  
 vtss\_phy\_wol\_conf\_get  
     vtss\_phy\_api.h, 992  
 vtss\_phy\_wol\_conf\_set  
     vtss\_phy\_api.h, 993  
 vtss\_phy\_wol\_conf\_t, 419  
     magic\_pkt\_cnt, 420  
     secure\_on\_enable, 419  
     wol\_mac, 419  
     wol\_pass, 420  
     wol\_passwd\_len, 420  
 vtss\_phy\_wol\_enable  
     vtss\_phy\_api.h, 992  
 vtss\_phy\_write  
     vtss\_phy\_api.h, 969  
 vtss\_phy\_write\_masked  
     vtss\_phy\_api.h, 970  
 vtss\_phy\_write\_masked\_page  
     vtss\_phy\_api.h, 970  
 vtss\_pi\_conf\_t, 420  
     cs\_wait\_ns, 421  
 vtss\_policer\_ext\_t, 421  
     broadcast, 422  
     cpu\_queue, 424  
     dp\_bypass\_level, 422  
     flooded, 423  
     flow\_control, 424  
     frame\_rate, 422  
     learning, 423  
     limit\_cpu\_traffic, 424  
     limit\_noncpu\_traffic, 424  
     mc\_no\_flood, 423  
     multicast, 422  
     to\_cpu, 423  
     uc\_no\_flood, 423  
     unicast, 422  
 vtss\_policer\_t, 425  
     level, 425  
     rate, 425  
 vtss\_policer\_type\_t  
     types.h, 626  
 vtss\_poll\_1sec  
     vtss\_misc\_api.h, 771  
 vtss\_port\_api.h  
     CHIP\_PORT\_UNUSED, 1077  
     VTSS\_FRAME\_GAP\_DEFAULT, 1077  
     VTSS\_MAX\_FRAME\_LENGTH\_MAX, 1078  
     VTSS\_MAX\_FRAME\_LENGTH\_STANDARD, 1078  
     vtss\_internal\_bw\_t, 1079  
     vtss\_miim\_controller\_t, 1078  
     vtss\_miim\_read, 1087  
     vtss\_miim\_write, 1087  
     vtss\_mmd\_read, 1090  
     vtss\_mmd\_write, 1091  
     vtss\_port\_basic\_counters\_get, 1085  
     vtss\_port\_clause\_37\_control\_get, 1081  
     vtss\_port\_clause\_37\_control\_set, 1082  
     vtss\_port\_clause\_37\_remote\_fault\_t, 1079  
     vtss\_port\_conf\_get, 1083  
     vtss\_port\_conf\_set, 1082  
     vtss\_port\_counters\_clear, 1084  
     vtss\_port\_counters\_get, 1084  
     vtss\_port\_counters\_update, 1083  
     vtss\_port\_forward\_state\_get, 1085  
     vtss\_port\_forward\_state\_set, 1086

vtss\_port\_forward\_t, 1080  
vtss\_port\_ifh\_conf\_get, 1086  
vtss\_port\_ifh\_conf\_set, 1086  
vtss\_port\_loop\_t, 1080  
vtss\_port\_map\_get, 1081  
vtss\_port\_map\_set, 1081  
vtss\_port\_max\_tags\_t, 1079  
vtss\_port\_mmd\_masked\_write, 1089  
vtss\_port\_mmd\_read, 1088  
vtss\_port\_mmd\_read\_inc, 1088  
vtss\_port\_mmd\_write, 1089  
vtss\_port\_status\_get, 1083  
vtss\_port\_basic\_counters\_get  
  vtss\_port\_api.h, 1085  
vtss\_port\_bridge\_counters\_t, 425  
  dot1dTpPortInDiscards, 426  
vtss\_port\_clause\_37\_adv\_t, 426  
  acknowledge, 427  
  asymmetric\_pause, 427  
  fdx, 427  
  hdx, 427  
  next\_page, 428  
  remote\_fault, 427  
  symmetric\_pause, 427  
vtss\_port\_clause\_37\_control\_get  
  vtss\_port\_api.h, 1081  
vtss\_port\_clause\_37\_control\_set  
  vtss\_port\_api.h, 1082  
vtss\_port\_clause\_37\_control\_t, 428  
  advertisement, 429  
  enable, 428  
vtss\_port\_clause\_37\_remote\_fault\_t  
  vtss\_port\_api.h, 1079  
vtss\_port\_conf\_get  
  vtss\_port\_api.h, 1083  
vtss\_port\_conf\_set  
  vtss\_port\_api.h, 1082  
vtss\_port\_conf\_t, 429  
  exc\_col\_cont, 432  
  fdx, 431  
  flow\_control, 431  
  frame\_gaps, 430  
  frame\_length\_chk, 432  
  if\_type, 430  
  loop, 433  
  max\_frame\_length, 431  
  max\_tags, 432  
  power\_down, 431  
  sd\_active\_high, 430  
  sd\_enable, 430  
  sd\_internal, 430  
  serdes, 433  
  speed, 431  
  xaui\_rx\_lane\_flip, 432  
  xaui\_tx\_lane\_flip, 432  
vtss\_port\_counters\_clear  
  vtss\_port\_api.h, 1084  
vtss\_port\_counters\_get  
  vtss\_port\_api.h, 1084  
vtss\_port\_counters\_t, 433  
  bridge, 434  
  ethernet\_like, 434  
  if\_group, 434  
  prop, 434  
  rmon, 434  
vtss\_port\_counters\_update  
  vtss\_port\_api.h, 1083  
vtss\_port\_ethernet\_like\_counters\_t, 435  
  dot3ControlInUnknownOpcodes, 436  
  dot3InPauseFrames, 436  
  dot3OutPauseFrames, 438  
  dot3StatsAlignmentErrors, 435  
  dot3StatsCarrierSenseErrors, 438  
  dot3StatsDeferredTransmissions, 437  
  dot3StatsExcessiveCollisions, 437  
  dot3StatsFCSErrors, 436  
  dot3StatsFrameTooLongs, 436  
  dot3StatsLateCollisions, 437  
  dot3StatsMultipleCollisionFrames, 437  
  dot3StatsSingleCollisionFrames, 437  
  dot3StatsSymbolErrors, 436  
vtss\_port\_flow\_control\_conf\_t, 438  
  generate, 439  
  obey, 439  
  pfc, 439  
  smac, 439  
vtss\_port\_forward\_state\_get  
  vtss\_port\_api.h, 1085  
vtss\_port\_forward\_state\_set  
  vtss\_port\_api.h, 1086  
vtss\_port\_forward\_t  
  vtss\_port\_api.h, 1080  
vtss\_port\_frame\_gaps\_t, 440  
  fdx\_gap, 440  
  hdx\_gap\_1, 440  
  hdx\_gap\_2, 440  
vtss\_port\_if\_group\_counters\_t, 441  
  ifInBroadcastPkts, 442  
  ifInDiscards, 442  
  ifInErrors, 443  
  ifInMulticastPkts, 442  
  ifInNUcastPkts, 442  
  ifInOctets, 441  
  ifInUcastPkts, 442  
  ifOutBroadcastPkts, 443  
  ifOutDiscards, 444  
  ifOutErrors, 444  
  ifOutMulticastPkts, 443  
  ifOutNUcastPkts, 444  
  ifOutOctets, 443  
  ifOutUcastPkts, 443  
vtss\_port\_ifh\_conf\_get  
  vtss\_port\_api.h, 1086  
vtss\_port\_ifh\_conf\_set  
  vtss\_port\_api.h, 1086  
vtss\_port\_ifh\_t, 444

ena\_ifh\_header, 445  
 vtss\_port\_interface\_t  
     types.h, 624  
 vtss\_port\_loop\_t  
     vtss\_port\_api.h, 1080  
 vtss\_port\_map\_get  
     vtss\_port\_api.h, 1081  
 vtss\_port\_map\_set  
     vtss\_port\_api.h, 1081  
 vtss\_port\_map\_t, 445  
     chip\_no, 446  
     chip\_port, 446  
     miim\_addr, 446  
     miim\_chip\_no, 446  
     miim\_controller, 446  
 vtss\_port\_max\_tags\_t  
     vtss\_port\_api.h, 1079  
 vtss\_port\_mmd\_masked\_write  
     vtss\_port\_api.h, 1089  
 vtss\_port\_mmd\_read  
     vtss\_port\_api.h, 1088  
 vtss\_port\_mmd\_read\_inc  
     vtss\_port\_api.h, 1088  
 vtss\_port\_mmd\_write  
     vtss\_port\_api.h, 1089  
 vtss\_port\_mux\_mode\_t  
     vtss\_init\_api.h, 670  
 vtss\_port\_proprietary\_counters\_t, 447  
     rx\_prio, 447  
     tx\_prio, 447  
 vtss\_port\_rmon\_counters\_t, 448  
     rx\_etherStatsBroadcastPkts, 449  
     rx\_etherStatsCRCAlignErrors, 450  
     rx\_etherStatsDropEvents, 449  
     rx\_etherStatsFragments, 450  
     rx\_etherStatsJabbers, 450  
     rx\_etherStatsMulticastPkts, 449  
     rx\_etherStatsOctets, 449  
     rx\_etherStatsOversizePkts, 450  
     rx\_etherStatsPkts, 449  
     rx\_etherStatsPkts1024to1518Octets, 452  
     rx\_etherStatsPkts128to255Octets, 451  
     rx\_etherStatsPkts1519toMaxOctets, 452  
     rx\_etherStatsPkts256to511Octets, 451  
     rx\_etherStatsPkts512to1023Octets, 451  
     rx\_etherStatsPkts64Octets, 451  
     rx\_etherStatsPkts65to127Octets, 451  
     rx\_etherStatsUndersizePkts, 450  
     tx\_etherStatsBroadcastPkts, 453  
     tx\_etherStatsCollisions, 453  
     tx\_etherStatsDropEvents, 452  
     tx\_etherStatsMulticastPkts, 453  
     tx\_etherStatsOctets, 452  
     tx\_etherStatsPkts, 452  
     tx\_etherStatsPkts1024to1518Octets, 454  
     tx\_etherStatsPkts128to255Octets, 454  
     tx\_etherStatsPkts1519toMaxOctets, 454  
     tx\_etherStatsPkts256to511Octets, 454  
 tx\_etherStatsPkts512to1023Octets, 454  
 tx\_etherStatsPkts64Octets, 453  
 tx\_etherStatsPkts65to127Octets, 453  
 tx\_etherStatsPkts1024to1518Octets, 454  
 tx\_etherStatsPkts128to255Octets, 454  
 tx\_etherStatsPkts1519toMaxOctets, 454  
 tx\_etherStatsPkts256to511Octets, 454  
 tx\_etherStatsPkts65to127Octets, 454  
 tx\_etherStatsPkts1024to1518Octets, 454  
 tx\_etherStatsPkts128to255Octets, 454  
 tx\_etherStatsPkts1519toMaxOctets, 454  
 tx\_etherStatsPkts256to511Octets, 454  
 sfp\_dac, 455  
 vtss\_port\_sgmii\_aneg\_t, 456  
     aneg\_complete, 457  
     fdx, 456  
     hdx, 456  
     link, 456  
     speed\_100M, 457  
     speed\_10M, 457  
     speed\_1G, 457  
 vtss\_port\_speed\_t  
     port.h, 591  
 vtss\_port\_state\_get  
     vtss\_l2\_api.h, 704  
 vtss\_port\_state\_set  
     vtss\_l2\_api.h, 705  
 vtss\_port\_status\_get  
     vtss\_port\_api.h, 1083  
 vtss\_port\_status\_t, 458  
     aneg, 459  
     aneg\_complete, 459  
     copper, 460  
     fdx, 459  
     fiber, 460  
     link, 458  
     link\_down, 458  
     mdi\_cross, 460  
     remote\_fault, 459  
     speed, 458  
     unidirectional\_ability, 459  
 vtss\_ptp\_event\_enable  
     vtss\_misc\_api.h, 771  
 vtss\_ptp\_event\_poll  
     vtss\_misc\_api.h, 771  
 vtss\_pvlan\_port\_members\_get  
     vtss\_l2\_api.h, 719  
 vtss\_pvlan\_port\_members\_set  
     vtss\_l2\_api.h, 719  
 vtss\_qce\_action\_t, 460  
     dp, 461  
     dp\_enable, 461  
     dscp, 462  
     dscp\_enable, 462  
     prio, 461  
     prio\_enable, 461  
 vtss\_qce\_add  
     vtss\_qos\_api.h, 1098  
 vtss\_qce\_del  
     vtss\_qos\_api.h, 1099  
 vtss\_qce\_frame\_etype\_t, 462  
     data, 463  
     etype, 463  
 vtss\_qce\_frame\_ipv4\_t, 463  
     dport, 464  
     dscp, 464

fragment, 464  
proto, 464  
sip, 464  
sport, 464  
vtss\_qce\_frame\_ipv6\_t, 465  
dport, 466  
dscp, 465  
proto, 465  
sip, 466  
sport, 466  
vtss\_qce\_frame\_llc\_t, 466  
data, 467  
vtss\_qce\_frame\_snap\_t, 467  
data, 467  
vtss\_qce\_init  
  vtss\_qos\_api.h, 1098  
vtss\_qce\_key\_t, 468  
  etype, 469  
  frame, 470  
  ipv4, 470  
  ipv6, 470  
  llc, 469  
  mac, 468  
  port\_list, 468  
  snap, 469  
  tag, 469  
  type, 469  
vtss\_qce\_mac\_t, 470  
  dmac\_bc, 471  
  dmac\_mc, 471  
  smac, 471  
vtss\_qce\_t, 472  
  action, 472  
  id, 472  
  key, 472  
vtss\_qce\_tag\_t, 473  
  dei, 474  
  pcp, 473  
  tagged, 474  
  vid, 473  
vtss\_qce\_type\_t  
  vtss\_qos\_api.h, 1096  
vtss\_qos\_api.h  
  VTSS\_PORT\_POLICER\_CPU\_QUEUES, 1094  
  VTSS\_PORT\_POLICERS, 1093  
  VTSS\_QCE\_ID\_LAST, 1095  
  VTSS\_QCL\_ARRAY\_SIZE, 1094  
  VTSS\_QCL\_ID\_END, 1094  
  VTSS\_QCL\_ID\_START, 1094  
  VTSS\_QCL\_IDS, 1094  
  vtss\_dscp\_emode\_t, 1096  
  vtss\_dscp\_mode\_t, 1095  
  vtss\_qce\_add, 1098  
  vtss\_qce\_del, 1099  
  vtss\_qce\_init, 1098  
  vtss\_qce\_type\_t, 1096  
  vtss\_qos\_conf\_get, 1096  
  vtss\_qos\_conf\_set, 1097  
vtss\_qos\_port\_conf\_get, 1097  
vtss\_qos\_port\_conf\_set, 1098  
vtss\_tag\_remark\_mode\_t, 1095  
vtss\_qos\_conf\_get  
  vtss\_qos\_api.h, 1096  
vtss\_qos\_conf\_set  
  vtss\_qos\_api.h, 1097  
vtss\_qos\_conf\_t, 474  
  dscp\_dp\_level\_map, 475  
  dscp\_qos\_class\_map, 475  
  dscp\_qos\_map, 475  
  dscp\_remap, 476  
  dscp\_remark, 476  
  dscp\_translate\_map, 476  
  dscp\_trust, 475  
  prios, 475  
vtss\_qos\_port\_conf\_get  
  vtss\_qos\_api.h, 1097  
vtss\_qos\_port\_conf\_set  
  vtss\_qos\_api.h, 1098  
vtss\_qos\_port\_conf\_t, 476  
  default\_dei, 479  
  default\_dpl, 479  
  default\_prio, 479  
  dp\_level\_map, 480  
  dscp\_class\_enable, 480  
  dscp\_emode, 480  
  dscp\_mode, 480  
  dscp\_translate, 481  
  dwrr\_enable, 482  
  excess\_enable, 478  
  policer\_ext\_port, 478  
  policer\_port, 477  
  policer\_queue, 478  
  qos\_class\_map, 480  
  queue\_pct, 482  
  red, 477  
  shaper\_port, 478  
  shaper\_queue, 478  
  tag\_class\_enable, 479  
  tag\_default\_dei, 481  
  tag\_default\_pcp, 481  
  tag\_dei\_map, 482  
  tag\_pcp\_map, 481  
  tag\_remark\_mode, 481  
  usr\_prio, 479  
vtss\_qs\_conf\_get  
  vtss\_init\_api.h, 675  
vtss\_qs\_conf\_set  
  vtss\_init\_api.h, 675  
vtss\_qs\_conf\_t, 482  
  mode, 483  
  oversubscription, 483  
  port, 484  
  port\_max, 483  
  port\_min, 483  
  queue\_max, 484  
  queue\_min, 484

vtss\_qs\_mode\_t  
     vtss\_init\_api.h, 670  
 vtss\_rcpll\_status\_t, 484  
     cal\_error, 485  
     cal\_not\_done, 485  
     out\_of\_range, 485  
 vtss\_recvrd\_clkout\_t  
     vtss\_phy\_10g\_api.h, 867  
 vtss\_recvrd\_t  
     vtss\_phy\_10g\_api.h, 863  
 vtss\_recvrdclk\_cdr\_div\_t  
     vtss\_phy\_10g\_api.h, 863  
 vtss\_red\_t, 486  
     enable, 486  
     max\_prob\_1, 487  
     max\_prob\_2, 487  
     max\_prob\_3, 487  
     max\_th, 486  
     min\_th, 486  
 vtss\_reg\_read  
     vtss\_misc\_api.h, 768  
 vtss\_reg\_read\_t  
     vtss\_init\_api.h, 663  
 vtss\_reg\_write  
     vtss\_misc\_api.h, 769  
 vtss\_reg\_write\_masked  
     vtss\_misc\_api.h, 769  
 vtss\_reg\_write\_t  
     vtss\_init\_api.h, 663  
 vtss\_restart\_conf\_end  
     vtss\_init\_api.h, 673  
 vtss\_restart\_conf\_get  
     vtss\_init\_api.h, 674  
 vtss\_restart\_conf\_set  
     vtss\_init\_api.h, 675  
 vtss\_restart\_status\_get  
     vtss\_init\_api.h, 674  
 vtss\_restart\_status\_t, 487  
     cur\_version, 488  
     prev\_version, 488  
     restart, 488  
 vtss\_restart\_t  
     vtss\_init\_api.h, 670  
 vtss\_routing\_entry\_t, 489  
     ipv4\_uc, 489  
     ipv6\_uc, 489  
     route, 490  
     type, 489  
     vlan, 490  
 vtss\_rptr\_rate\_t  
     vtss\_phy\_10g\_api.h, 865  
 vtss\_rx\_master\_timestamp\_get  
     vtss\_ts\_api.h, 1129  
 vtss\_rx\_timestamp\_get  
     vtss\_ts\_api.h, 1128  
 vtss\_rx\_timestamp\_id\_release  
     vtss\_ts\_api.h, 1129  
 vtss\_secure\_on\_passwd\_t, 490  
     passwd, 491  
 vtss\_security\_api.h  
     VTSS\_ACE\_ID\_LAST, 1102  
     VTSS\_PORT\_NO\_ANY, 1102  
     vtss\_ace\_add, 1107  
     vtss\_ace\_bit\_t, 1103  
     vtss\_ace\_counter\_clear, 1109  
     vtss\_ace\_counter\_get, 1108  
     vtss\_ace\_del, 1108  
     vtss\_ace\_init, 1107  
     vtss\_ace\_type\_t, 1103  
     vtss\_acl\_policer\_conf\_get, 1104  
     vtss\_acl\_policer\_conf\_set, 1105  
     vtss\_acl\_port\_conf\_get, 1105  
     vtss\_acl\_port\_conf\_set, 1106  
     vtss\_acl\_port\_counter\_clear, 1106  
     vtss\_acl\_port\_counter\_get, 1106  
     vtss\_auth\_port\_state\_get, 1103  
     vtss\_auth\_port\_state\_set, 1104  
     vtss\_auth\_state\_t, 1102  
 vtss\_serdes\_macro\_conf\_t, 491  
     ib\_cterm\_ena, 492  
     qsgmii, 492  
     serdes1g\_vdd, 491  
     serdes6g\_vdd, 491  
 vtss\_serdes\_mode\_t  
     types.h, 625  
 vtss\_sfflow\_port\_conf\_get  
     vtss\_l2\_api.h, 721  
 vtss\_sfflow\_port\_conf\_set  
     vtss\_l2\_api.h, 722  
 vtss\_sfflow\_port\_conf\_t, 492  
     sampling\_rate, 493  
     type, 493  
 vtss\_sfflow\_sampling\_rate\_convert  
     vtss\_l2\_api.h, 722  
 vtss\_sfflow\_type\_t  
     l2\_types.h, 557  
 vtss\_sgpio\_bmode\_t  
     vtss\_misc\_api.h, 763  
 vtss\_sgpio\_conf\_get  
     vtss\_misc\_api.h, 776  
 vtss\_sgpio\_conf\_set  
     vtss\_misc\_api.h, 776  
 vtss\_sgpio\_conf\_t, 493  
     bit\_count, 494  
     bmode, 494  
     port\_conf, 494  
 vtss\_sgpio\_event\_enable  
     vtss\_misc\_api.h, 777  
 vtss\_sgpio\_event\_poll  
     vtss\_misc\_api.h, 777  
 vtss\_sgpio\_mode\_t  
     vtss\_misc\_api.h, 763  
 vtss\_sgpio\_port\_conf\_t, 494  
     enabled, 495  
     int\_pol\_high, 495  
     mode, 495

vtss\_sgpi\_port\_data\_t, 496  
    value, 496  
vtss\_sgpi\_read  
    vtss\_misc\_api.h, 776  
vtss\_shaper\_t, 496  
    level, 497  
    rate, 497  
vtss\_spi\_32bit\_read\_write\_t  
    vtss\_init\_api.h, 665  
vtss\_spi\_64bit\_read\_write\_t  
    vtss\_init\_api.h, 666  
vtss\_spi\_read\_write\_t  
    vtss\_init\_api.h, 665  
vtss\_squelch\_workaround  
    vtss\_phy\_api.h, 980  
vtss\_srefclk\_div\_t  
    vtss\_phy\_10g\_api.h, 864  
vtss\_storm\_policer\_mode\_t  
    types.h, 626  
vtss\_stp\_port\_state\_get  
    vtss\_l2\_api.h, 705  
vtss\_stp\_port\_state\_set  
    vtss\_l2\_api.h, 705  
vtss\_stp\_state\_t  
    vtss\_l2\_api.h, 692  
vtss\_sublayer\_status\_t, 497  
    link\_down, 498  
    rx\_fault, 498  
    rx\_link, 498  
    tx\_fault, 498  
vtss\_sync\_api.h  
    VTSS\_SYNCE\_CLK\_MAX, 1111  
    VTSS\_SYNCE\_CLK\_A, 1110  
    VTSS\_SYNCE\_CLK\_B, 1110  
    vtss\_sync\_clock\_in\_get, 1113  
    vtss\_sync\_clock\_in\_set, 1112  
    vtss\_sync\_clock\_out\_get, 1112  
    vtss\_sync\_clock\_out\_set, 1111  
    vtss\_sync\_divider\_t, 1111  
vtss\_sync\_clock\_in\_get  
    vtss\_sync\_api.h, 1113  
vtss\_sync\_clock\_in\_set  
    vtss\_sync\_api.h, 1112  
vtss\_sync\_clock\_in\_t, 499  
    enable, 499  
    port\_no, 499  
    squels, 499  
vtss\_sync\_clock\_out\_get  
    vtss\_sync\_api.h, 1112  
vtss\_sync\_clock\_out\_set  
    vtss\_sync\_api.h, 1111  
vtss\_sync\_clock\_out\_t, 500  
    divider, 500  
    enable, 500  
vtss\_sync\_divider\_t  
    vtss\_sync\_api.h, 1111  
vtss\_tag\_remark\_mode\_t  
    vtss\_qos\_api.h, 1095  
vtss\_tag\_type\_t  
    vtss\_packet\_api.h, 816  
vtss\_target\_type\_t  
    vtss\_init\_api.h, 669  
vtss\_tci\_t, 501  
    cfi, 501  
    tagprio, 502  
    vid, 501  
vtss\_temp\_sensor\_get  
    vtss\_misc\_api.h, 781  
vtss\_temp\_sensor\_init  
    vtss\_misc\_api.h, 781  
vtss\_timeofday\_t, 502  
    sec, 502, 503  
vtss\_timestamp\_age  
    vtss\_ts\_api.h, 1130  
vtss\_timestamp\_t, 503  
    nanoseconds, 504  
    sec\_msb, 503  
    seconds, 504  
vtss\_tod\_get\_ns\_cnt  
    vtss\_misc\_api.h, 780  
vtss\_tod\_set\_ns\_cnt\_cb  
    vtss\_misc\_api.h, 780  
vtss\_trace\_conf\_get  
    vtss\_misc\_api.h, 764  
vtss\_trace\_conf\_set  
    vtss\_misc\_api.h, 765  
vtss\_trace\_conf\_t, 504  
    level, 505  
vtss\_trace\_group\_t  
    vtss\_misc\_api.h, 760  
vtss\_trace\_layer\_t  
    vtss\_misc\_api.h, 759  
vtss\_trace\_level\_t  
    vtss\_misc\_api.h, 760  
vtss\_ts\_adjtimer\_get  
    vtss\_ts\_api.h, 1121  
vtss\_ts\_adjtimer\_one\_sec  
    vtss\_ts\_api.h, 1119  
vtss\_ts\_adjtimer\_set  
    vtss\_ts\_api.h, 1120  
vtss\_ts\_api.h  
    vtss\_rx\_master\_timestamp\_get, 1129  
    vtss\_rx\_timestamp\_get, 1128  
    vtss\_rx\_timestamp\_id\_release, 1129  
    vtss\_timestamp\_age, 1130  
    vtss\_ts\_adjtimer\_get, 1121  
    vtss\_ts\_adjtimer\_one\_sec, 1119  
    vtss\_ts\_adjtimer\_set, 1120  
    vtss\_ts\_delay\_asymmetry\_get, 1126  
    vtss\_ts\_delay\_asymmetry\_set, 1125  
    vtss\_ts\_egress\_latency\_get, 1125  
    vtss\_ts\_egress\_latency\_set, 1124  
    vtss\_ts\_external\_clock\_mode\_get, 1121  
    vtss\_ts\_external\_clock\_mode\_set, 1122  
    vtss\_ts\_external\_clock\_saved\_get, 1122  
    vtss\_ts\_freq\_offset\_get, 1121

vtss\_ts\_ingress\_latency\_get, 1123  
 vtss\_ts\_ingress\_latency\_set, 1123  
 vtss\_ts\_internal\_mode\_get, 1127  
 vtss\_ts\_internal\_mode\_set, 1127  
 vtss\_ts\_ongoing\_adjustment, 1119  
 vtss\_ts\_operation\_mode\_get, 1127  
 vtss\_ts\_operation\_mode\_set, 1126  
 vtss\_ts\_p2p\_delay\_get, 1124  
 vtss\_ts\_p2p\_delay\_set, 1124  
 vtss\_ts\_status\_change, 1130  
 vtss\_ts\_timeofday\_get, 1120  
 vtss\_ts\_timeofday\_next\_pps\_get, 1120  
 vtss\_ts\_timeofday\_offset\_set, 1117  
 vtss\_ts\_timeofday\_set, 1117  
 vtss\_ts\_timeofday\_set\_delta, 1117  
 vtss\_tx\_timestamp\_idx\_alloc, 1129  
 vtss\_tx\_timestamp\_update, 1128  
 vtss\_ts\_delay\_asymmetry\_get  
     vtss\_ts\_api.h, 1126  
 vtss\_ts\_delay\_asymmetry\_set  
     vtss\_ts\_api.h, 1125  
 vtss\_ts\_egress\_latency\_get  
     vtss\_ts\_api.h, 1125  
 vtss\_ts\_egress\_latency\_set  
     vtss\_ts\_api.h, 1124  
 vtss\_ts\_ext\_clock\_mode\_t, 505  
     enable, 505  
     freq, 506  
     one\_pps\_mode, 505  
 vtss\_ts\_external\_clock\_mode\_get  
     vtss\_ts\_api.h, 1121  
 vtss\_ts\_external\_clock\_mode\_set  
     vtss\_ts\_api.h, 1122  
 vtss\_ts\_external\_clock\_saved\_get  
     vtss\_ts\_api.h, 1122  
 vtss\_ts\_freq\_offset\_get  
     vtss\_ts\_api.h, 1121  
 vtss\_ts\_id\_t, 506  
     ts\_id, 506  
 vtss\_ts\_ingress\_latency\_get  
     vtss\_ts\_api.h, 1123  
 vtss\_ts\_ingress\_latency\_set  
     vtss\_ts\_api.h, 1123  
 vtss\_ts\_internal\_mode\_get  
     vtss\_ts\_api.h, 1127  
 vtss\_ts\_internal\_mode\_set  
     vtss\_ts\_api.h, 1127  
 vtss\_ts\_internal\_mode\_t, 507  
     int\_fmt, 507  
 vtss\_ts\_ongoing\_adjustment  
     vtss\_ts\_api.h, 1119  
 vtss\_ts\_operation\_mode\_get  
     vtss\_ts\_api.h, 1127  
 vtss\_ts\_operation\_mode\_set  
     vtss\_ts\_api.h, 1126  
 vtss\_ts\_operation\_mode\_t, 508  
     mode, 508  
 vtss\_ts\_p2p\_delay\_get  
     vtss\_ts\_api.h, 1124  
 vtss\_ts\_p2p\_delay\_set  
     vtss\_ts\_api.h, 1124  
 vtss\_ts\_status\_change  
     vtss\_ts\_api.h, 1130  
 vtss\_ts\_timeofday\_get  
     vtss\_ts\_api.h, 1120  
 vtss\_ts\_timeofday\_next\_pps\_get  
     vtss\_ts\_api.h, 1120  
 vtss\_ts\_timeofday\_offset\_set  
     vtss\_ts\_api.h, 1117  
 vtss\_ts\_timeofday\_set  
     vtss\_ts\_api.h, 1117  
 vtss\_ts\_timeofday\_set\_delta  
     vtss\_ts\_api.h, 1117  
 vtss\_ts\_timestamp\_alloc\_t, 508  
     cb, 509  
     context, 509  
     port\_mask, 509  
 vtss\_ts\_timestamp\_t, 509  
     context, 510  
     id, 510  
     ts, 510  
     ts\_valid, 510  
 vtss\_tx\_timestamp\_idx\_alloc  
     vtss\_ts\_api.h, 1129  
 vtss\_tx\_timestamp\_update  
     vtss\_ts\_api.h, 1128  
 vtss\_uc\_flood\_members\_get  
     vtss\_l2\_api.h, 731  
 vtss\_uc\_flood\_members\_set  
     vtss\_l2\_api.h, 732  
 vtss\_vcap\_bit\_t  
     types.h, 627  
 vtss\_vcap\_ip\_t, 511  
     mask, 511  
     value, 511  
 vtss\_vcap\_key\_type\_t  
     types.h, 628  
 vtss\_vcap\_u128\_t, 512  
     mask, 512  
     value, 512  
 vtss\_vcap\_u16\_t, 513  
     mask, 513  
     value, 513  
 vtss\_vcap\_u24\_t, 514  
     mask, 514  
     value, 514  
 vtss\_vcap\_u32\_t, 515  
     mask, 515  
     value, 515  
 vtss\_vcap\_u40\_t, 516  
     mask, 516  
     value, 516  
 vtss\_vcap\_u48\_t, 517  
     mask, 517  
     value, 517  
 vtss\_vcap\_u8\_t, 518

mask, 518  
value, 518  
`vtss_vcap_udp_tcp_t`, 519  
high, 520  
in\_range, 519  
low, 519  
`vtss_vcap_vid_t`, 520  
mask, 521  
value, 520  
`vtss_vcap_vr_t`, 521  
high, 522  
low, 522  
mask, 522  
r, 523  
type, 522  
v, 522  
value, 522  
vr, 523  
`vtss_vcap_vr_type_t`  
types.h, 628  
`vtss_vce_action_t`, 523  
policy\_no, 524  
vid, 523  
`vtss_vce_add`  
vtss\_l2\_api.h, 714  
`vtss_vce_del`  
vtss\_l2\_api.h, 714  
`vtss_vce_frame_etype_t`, 524  
data, 525  
etype, 524  
`vtss_vce_frame_ipv4_t`, 525  
dport, 526  
dscp, 526  
fragment, 525  
options, 526  
proto, 526  
sip, 526  
`vtss_vce_frame_ipv6_t`, 527  
dport, 528  
dscp, 527  
proto, 527  
sip, 528  
`vtss_vce_frame_llc_t`, 528  
data, 529  
`vtss_vce_frame_snap_t`, 529  
data, 529  
`vtss_vce_init`  
vtss\_l2\_api.h, 714  
`vtss_vce_key_t`, 530  
etype, 531  
frame, 532  
ipv4, 531  
ipv6, 532  
llc, 531  
mac, 530  
port\_list, 530  
snap, 531  
tag, 530  
type, 531  
`vtss_vce_mac_t`, 532  
dmac\_bc, 533  
dmac\_mc, 533  
smac, 533  
`vtss_vce_t`, 533  
action, 534  
id, 534  
key, 534  
`vtss_vce_tag_t`, 534  
dei, 535  
pcp, 535  
s\_tag, 536  
tagged, 535  
vid, 535  
`vtss_vce_type_t`  
vtss\_l2\_api.h, 693  
`vtss_vcl_port_conf_get`  
vtss\_l2\_api.h, 713  
`vtss_vcl_port_conf_set`  
vtss\_l2\_api.h, 713  
`vtss_vcl_port_conf_t`, 536  
dmac\_dip, 536  
`vtss_vdd_t`  
types.h, 627  
`vtss_vid_mac_t`, 537  
mac, 537  
vid, 537  
`vtss_vlan_conf_get`  
vtss\_l2\_api.h, 708  
`vtss_vlan_conf_set`  
vtss\_l2\_api.h, 708  
`vtss_vlan_conf_t`, 538  
s\_etype, 538  
`vtss_vlan_frame_t`  
types.h, 626  
`vtss_vlan_port_conf_get`  
vtss\_l2\_api.h, 708  
`vtss_vlan_port_conf_set`  
vtss\_l2\_api.h, 710  
`vtss_vlan_port_conf_t`, 539  
frame\_type, 540  
ingress\_filter, 540  
port\_type, 539  
pvid, 539  
untagged\_vid, 539  
`vtss_vlan_port_members_get`  
vtss\_l2\_api.h, 710  
`vtss_vlan_port_members_set`  
vtss\_l2\_api.h, 711  
`vtss_vlan_port_type_t`  
vtss\_l2\_api.h, 693  
`vtss_vlan_tag_t`, 540  
dei, 541  
pcp, 541  
tpid, 541  
vid, 541  
`vtss_vlan_trans_group_add`

vtss\_l2\_api.h, 715  
 vtss\_vlan\_trans\_group\_del  
     vtss\_l2\_api.h, 715  
 vtss\_vlan\_trans\_group\_get  
     vtss\_l2\_api.h, 716  
 vtss\_vlan\_trans\_group\_to\_port\_get  
     vtss\_l2\_api.h, 717  
 vtss\_vlan\_trans\_group\_to\_port\_set  
     vtss\_l2\_api.h, 716  
 vtss\_vlan\_trans\_grp2vlan\_conf\_t, 542  
     group\_id, 542  
     trans\_vid, 542  
     vid, 542  
 vtss\_vlan\_trans\_port2grp\_conf\_t, 543  
     group\_id, 543  
     ports, 543  
 vtss\_vlan\_tx\_tag\_get  
     vtss\_l2\_api.h, 712  
 vtss\_vlan\_tx\_tag\_set  
     vtss\_l2\_api.h, 712  
 vtss\_vlan\_tx\_tag\_t  
     vtss\_l2\_api.h, 693  
 vtss\_vlan\_vid\_conf\_get  
     vtss\_l2\_api.h, 711  
 vtss\_vlan\_vid\_conf\_set  
     vtss\_l2\_api.h, 711  
 vtss\_vlan\_vid\_conf\_t, 544  
     fid, 545  
     learning, 544  
     mirror, 544  
 vtss\_vstax\_conf\_get  
     vtss\_l2\_api.h, 738  
 vtss\_vstax\_conf\_set  
     vtss\_l2\_api.h, 739  
 vtss\_vstax\_conf\_t, 545  
     cmeff\_disable, 546  
     port\_0, 546  
     port\_1, 546  
     upsid\_0, 545  
     upsid\_1, 546  
 vtss\_vstax\_fwd\_mode\_t  
     vtss\_packet\_api.h, 815  
 vtss\_vstax\_glag\_entry\_t, 547  
     upsid, 547  
     upspn, 547  
 vtss\_vstax\_glag\_get  
     vtss\_l2\_api.h, 726  
 vtss\_vstax\_glag\_set  
     vtss\_l2\_api.h, 726  
 vtss\_vstax\_master\_upsid\_get  
     vtss\_l2\_api.h, 740  
 vtss\_vstax\_master\_upsid\_set  
     vtss\_l2\_api.h, 741  
 vtss\_vstax\_port\_conf\_get  
     vtss\_l2\_api.h, 739  
 vtss\_vstax\_port\_conf\_set  
     vtss\_l2\_api.h, 740  
 vtss\_vstax\_port\_conf\_t, 547  
     mirror, 548  
     ttl, 548  
 vtss\_vstax\_route\_entry\_t, 548  
     stack\_port\_a, 549  
     stack\_port\_b, 549  
 vtss\_vstax\_route\_table\_t, 549  
     table, 550  
     topology\_type, 550  
 vtss\_vstax\_rx\_header\_t, 550  
     glag\_no, 551  
     isdx, 552  
     port\_no, 551  
     sp, 551  
     upsid, 551  
     valid, 551  
 vtss\_vstax\_topology\_set  
     vtss\_l2\_api.h, 741  
 vtss\_vstax\_topology\_type\_t  
     vtss\_l2\_api.h, 695  
 vtss\_vstax\_tx\_header\_t, 552  
     chip\_port, 554  
     dp, 554  
     fwd\_mode, 553  
     glag\_no, 554  
     keep\_ttl, 554  
     port\_no, 553  
     prio, 553  
     queue\_no, 554  
     tci, 553  
     ttl, 553  
     upsid, 553  
 vtss\_vt\_id\_t  
     vtss\_l2\_api.h, 692  
 vtss\_wis\_api.h  
     VTSS\_EWIS\_AISL\_EV, 1138  
     VTSS\_EWIS\_AISP\_EV, 1138  
     VTSS\_EWIS\_B1\_NZ\_EV, 1141  
     VTSS\_EWIS\_B1\_THRESH\_EV, 1142  
     VTSS\_EWIS\_B2\_NZ\_EV, 1141  
     VTSS\_EWIS\_B2\_THRESH\_EV, 1142  
     VTSS\_EWIS\_B3\_NZ\_EV, 1141  
     VTSS\_EWIS\_B3\_THRESH\_EV, 1142  
     VTSS\_EWIS\_FAIS\_EV, 1137  
     VTSS\_EWIS\_FERDIP\_EV, 1140  
     VTSS\_EWIS\_FEUNEQP\_EV, 1140  
     VTSS\_EWIS\_FPLM\_EV, 1137  
     VTSS\_EWIS\_HIGH\_BER\_EV, 1140  
     VTSS\_EWIS\_LCDP\_EV, 1138  
     VTSS\_EWIS\_LOF\_EV, 1137  
     VTSS\_EWIS\_LOPC\_EV, 1139  
     VTSS\_EWIS\_LOPP\_EV, 1138  
     VTSS\_EWIS\_LOS\_EV, 1137  
     VTSS\_EWIS\_MODULE\_EV, 1139  
     VTSS\_EWIS\_PCS\_RECEIVE\_FAULT\_PEND,  
         1141  
     VTSS\_EWIS\_PLMP\_EV, 1138  
     VTSS\_EWIS\_RDIL\_EV, 1137  
     VTSS\_EWIS\_REIL\_EV, 1140

VTSS\_EWIS\_REIL\_NZ\_EV, 1141  
VTSS\_EWIS\_REIL\_THRESH\_EV, 1142  
VTSS\_EWIS\_REIP\_EV, 1140  
VTSS\_EWIS\_REIP\_NZ\_EV, 1142  
VTSS\_EWIS\_REIP\_THRESH\_EV, 1143  
VTSS\_EWIS\_RXLOL\_EV, 1139  
VTSS\_EWIS\_SEF\_EV, 1136  
VTSS\_EWIS\_TXLOL\_EV, 1139  
VTSS\_EWIS\_UNEQP\_EV, 1139  
vtss\_ewis\_cons\_act\_get, 1152  
vtss\_ewis\_cons\_act\_set, 1152  
vtss\_ewis\_counter\_get, 1160  
vtss\_ewis\_counter\_threshold\_get, 1161  
vtss\_ewis\_counter\_threshold\_set, 1160  
vtss\_ewis\_defects\_get, 1158  
vtss\_ewis\_event\_enable, 1145  
vtss\_ewis\_event\_force, 1147  
vtss\_ewis\_event\_poll, 1146  
vtss\_ewis\_event\_poll\_without\_mask, 1146  
vtss\_ewis\_event\_t, 1143  
vtss\_ewis\_exp\_sl\_set, 1153  
vtss\_ewis\_force\_conf\_get, 1148  
vtss\_ewis\_force\_conf\_set, 1148  
vtss\_ewis\_mode\_get, 1151  
vtss\_ewis\_mode\_set, 1150  
vtss\_ewis\_mode\_t, 1144  
vtss\_ewis\_path\_acti\_get, 1159  
vtss\_ewis\_path\_txti\_get, 1154  
vtss\_ewis\_path\_txti\_set, 1154  
vtss\_ewis\_perf\_cntr\_mode\_t, 1144  
vtss\_ewis\_perf\_get, 1160  
vtss\_ewis\_perf\_mode\_get, 1163  
vtss\_ewis\_perf\_mode\_set, 1161  
vtss\_ewis\_prbs31\_err\_inj\_set, 1157  
vtss\_ewis\_prbs31\_err\_inj\_t, 1145  
vtss\_ewis\_reset, 1151  
vtss\_ewis\_section\_acti\_get, 1159  
vtss\_ewis\_section\_txti\_get, 1153  
vtss\_ewis\_section\_txti\_set, 1153  
vtss\_ewis\_static\_conf\_get, 1147  
vtss\_ewis\_static\_conf\_t, 1143  
vtss\_ewis\_status\_get, 1158  
vtss\_ewis\_test\_counter\_get, 1157  
vtss\_ewis\_test\_mode\_get, 1156  
vtss\_ewis\_test\_mode\_set, 1156  
vtss\_ewis\_test\_pattern\_s, 1144  
vtss\_ewis\_tti\_mode\_t, 1143  
vtss\_ewis\_tx\_oh\_get, 1149  
vtss\_ewis\_tx\_oh\_passthru\_get, 1150  
vtss\_ewis\_tx\_oh\_passthru\_set, 1149  
vtss\_ewis\_tx\_oh\_set, 1149  
vtss\_wol\_mac\_addr\_t, 555  
addr, 555  
vtss\_wol\_passwd\_len\_type\_t  
vtss\_phy\_api.h, 957  
vtss\_wref\_clk\_div\_t  
vtss\_phy\_10g\_api.h, 864  
vtss\_wrefclk\_t  
vtss\_phy\_10g\_api.h, 862  
warm\_start\_enable  
vtss\_init\_conf\_t, 162  
wis  
vtss\_phy\_10g\_serdes\_status\_t, 319  
vtss\_phy\_10g\_status\_t, 321  
wol\_mac  
vtss\_phy\_wol\_conf\_t, 419  
wol\_pass  
vtss\_phy\_wol\_conf\_t, 420  
wol\_passwd\_len  
vtss\_phy\_wol\_conf\_t, 420  
wref\_clk\_div  
vtss\_phy\_10g\_mode\_t, 284  
wrefclk  
vtss\_phy\_10g\_mode\_t, 282  
x\_count  
vtss\_phy\_10g\_vscope\_scan\_conf\_t, 327  
x\_incr  
vtss\_phy\_10g\_vscope\_scan\_conf\_t, 327  
x\_start  
vtss\_phy\_10g\_vscope\_scan\_conf\_t, 326  
xaui\_lane\_flip  
vtss\_phy\_10g\_mode\_t, 283  
xaui\_rx\_lane\_flip  
vtss\_port\_conf\_t, 432  
xaui\_sel\_8487  
vtss\_phy\_ts\_init\_conf\_t, 389  
xaui\_tx\_lane\_flip  
vtss\_port\_conf\_t, 432  
xfi\_pol\_invert  
vtss\_phy\_10g\_mode\_t, 283  
xs  
vtss\_phy\_10g\_status\_t, 322  
xtr\_cb  
vtss\_fdma\_ch\_cfg\_t, 149  
xtr\_grp  
vtss\_fdma\_ch\_cfg\_t, 148  
xtr\_qu  
vtss\_packet\_rx\_meta\_t, 212  
xtr\_qu\_mask  
vtss\_packet\_rx\_info\_t, 205  
y1731\_en  
vtss\_phy\_ts\_oam\_engine\_action\_t, 401  
y1731\_oam\_conf  
vtss\_phy\_ts\_oam\_engine\_action\_t, 401  
y\_count  
vtss\_phy\_10g\_vscope\_scan\_conf\_t, 327  
y\_incr  
vtss\_phy\_10g\_vscope\_scan\_conf\_t, 327  
y\_start  
vtss\_phy\_10g\_vscope\_scan\_conf\_t, 327