

# Vitesse API

Generated by Doxygen 1.8.14



# Contents

<b>1</b>	<b>Layer 2</b>	<b>1</b>
1.1	MAC Address Table . . . . .	1
1.2	Operational State . . . . .	2
1.3	Spanning Tree . . . . .	2
1.4	VLAN . . . . .	2
1.5	VLAN Classification List . . . . .	3
1.6	VLAN Translation . . . . .	3
1.7	Port Isolation . . . . .	4
1.8	Private VLAN . . . . .	4
1.9	Asymmetric Private VLAN . . . . .	4
1.10	Destination Port Groups . . . . .	4
1.11	sFlow . . . . .	5
1.12	Link Aggregation . . . . .	5
1.13	Global Link Aggregation . . . . .	5
1.14	Mirroring . . . . .	6
1.15	Flooding Control . . . . .	6
1.16	IPv4 Multicast . . . . .	6
1.17	IPv6 Multicast . . . . .	7
1.18	Ethernet Protection Switching . . . . .	7
1.19	Ethernet Ring Protection Switching . . . . .	7
1.20	VStaX . . . . .	7

<b>2 Security</b>	<b>9</b>
2.1 Port Authentication (802.1X) . . . . .	9
2.2 Access Control List . . . . .	9
2.2.1 Access Control Entry . . . . .	9
2.2.2 Port Configuration . . . . .	10
2.2.3 Policer Configuration . . . . .	10
<b>3 Data Structure Index</b>	<b>11</b>
3.1 Data Structures . . . . .	11
<b>4 File Index</b>	<b>23</b>
4.1 File List . . . . .	23
<b>5 Data Structure Documentation</b>	<b>25</b>
5.1 ib_par_cfg Struct Reference . . . . .	25
5.1.1 Detailed Description . . . . .	25
5.1.2 Field Documentation . . . . .	25
5.1.2.1 value . . . . .	25
5.1.2.2 min . . . . .	26
5.1.2.3 max . . . . .	26
5.2 port_custom_conf_t Struct Reference . . . . .	26
5.2.1 Detailed Description . . . . .	26
5.2.2 Field Documentation . . . . .	27
5.2.2.1 enable . . . . .	27
5.2.2.2 autoneg . . . . .	27
5.2.2.3 fdx . . . . .	27
5.2.2.4 flow_control . . . . .	27
5.2.2.5 pfc . . . . .	27
5.2.2.6 speed . . . . .	28
5.2.2.7 dual_media_fiber_speed . . . . .	28
5.2.2.8 max_length . . . . .	28
5.2.2.9 power_mode . . . . .	28

5.2.2.10	exc_col_cont . . . . .	28
5.2.2.11	adv_dis . . . . .	29
5.2.2.12	max_tags . . . . .	29
5.2.2.13	oper_up . . . . .	29
5.2.2.14	frame_length_chk . . . . .	29
5.3	serdes_fields_t Struct Reference . . . . .	29
5.3.1	Detailed Description . . . . .	30
5.3.2	Field Documentation . . . . .	30
5.3.2.1	ob_post0 . . . . .	30
5.3.2.2	ob_sr . . . . .	30
5.4	tag_vtss_fdma_list Struct Reference . . . . .	30
5.4.1	Detailed Description . . . . .	31
5.4.2	Field Documentation . . . . .	31
5.4.2.1	frm_ptr . . . . .	31
5.4.2.2	act_len . . . . .	32
5.4.2.3	alloc_ptr . . . . .	32
5.4.2.4	rx_info . . . . .	32
5.4.2.5	sw_tstamp . . . . .	32
5.4.2.6	user . . . . .	33
5.4.2.7	next . . . . .	33
5.5	vtss_ace_frame_arp_t Struct Reference . . . . .	33
5.5.1	Detailed Description . . . . .	34
5.5.2	Field Documentation . . . . .	34
5.5.2.1	smac . . . . .	34
5.5.2.2	arp . . . . .	34
5.5.2.3	req . . . . .	34
5.5.2.4	unknown . . . . .	34
5.5.2.5	smac_match . . . . .	35
5.5.2.6	dmac_match . . . . .	35
5.5.2.7	length . . . . .	35

5.5.2.8	ip	35
5.5.2.9	ethernet	35
5.5.2.10	sip	36
5.5.2.11	dip	36
5.6	vtss_ace_frame_etype_t Struct Reference	36
5.6.1	Detailed Description	36
5.6.2	Field Documentation	36
5.6.2.1	dmac	37
5.6.2.2	smac	37
5.6.2.3	etype	37
5.6.2.4	data	37
5.6.2.5	ptp	37
5.7	vtss_ace_frame_ipv4_t Struct Reference	38
5.7.1	Detailed Description	38
5.7.2	Field Documentation	38
5.7.2.1	ttl	38
5.7.2.2	fragment	39
5.7.2.3	options	39
5.7.2.4	ds	39
5.7.2.5	proto	39
5.7.2.6	sip	39
5.7.2.7	dip	40
5.7.2.8	data	40
5.7.2.9	sport	40
5.7.2.10	dport	40
5.7.2.11	tcp_fin	40
5.7.2.12	tcp_syn	41
5.7.2.13	tcp_RST	41
5.7.2.14	tcp_psh	41
5.7.2.15	tcp_ack	41

5.7.2.16	tcp_urg	41
5.7.2.17	sip_eq_dip	42
5.7.2.18	sport_eq_dport	42
5.7.2.19	seq_zero	42
5.7.2.20	ptp	42
5.7.2.21	sip_smac	42
5.8	vtss_ace_frame_ipv6_t Struct Reference	43
5.8.1	Detailed Description	43
5.8.2	Field Documentation	43
5.8.2.1	proto	43
5.8.2.2	sip	44
5.8.2.3	ttl	44
5.8.2.4	ds	44
5.8.2.5	data	44
5.8.2.6	sport	44
5.8.2.7	dport	45
5.8.2.8	tcp_fin	45
5.8.2.9	tcp_syn	45
5.8.2.10	tcp_RST	45
5.8.2.11	tcp_psh	45
5.8.2.12	tcp_ack	46
5.8.2.13	tcp_urg	46
5.8.2.14	sip_eq_dip	46
5.8.2.15	sport_eq_dport	46
5.8.2.16	seq_zero	46
5.8.2.17	ptp	47
5.9	vtss_ace_frame_llc_t Struct Reference	47
5.9.1	Detailed Description	47
5.9.2	Field Documentation	47
5.9.2.1	dmac	47

---

5.9.2.2	smac . . . . .	48
5.9.2.3	llc . . . . .	48
5.10	vtss_ace_frame_snap_t Struct Reference . . . . .	48
5.10.1	Detailed Description . . . . .	48
5.10.2	Field Documentation . . . . .	48
5.10.2.1	dmac . . . . .	49
5.10.2.2	smac . . . . .	49
5.10.2.3	snap . . . . .	49
5.11	vtss_ace_ptp_t Struct Reference . . . . .	49
5.11.1	Detailed Description . . . . .	49
5.11.2	Field Documentation . . . . .	50
5.11.2.1	enable . . . . .	50
5.11.2.2	header . . . . .	50
5.12	vtss_ace_sip_smac_t Struct Reference . . . . .	50
5.12.1	Detailed Description . . . . .	50
5.12.2	Field Documentation . . . . .	51
5.12.2.1	enable . . . . .	51
5.12.2.2	sip . . . . .	51
5.12.2.3	smac . . . . .	51
5.13	vtss_ace_t Struct Reference . . . . .	51
5.13.1	Detailed Description . . . . .	52
5.13.2	Field Documentation . . . . .	52
5.13.2.1	id . . . . .	52
5.13.2.2	port_list . . . . .	52
5.13.2.3	policy . . . . .	53
5.13.2.4	type . . . . .	53
5.13.2.5	action . . . . .	53
5.13.2.6	dmac_mc . . . . .	53
5.13.2.7	dmac_bc . . . . .	53
5.13.2.8	vlan . . . . .	54

5.13.2.9 <code>etype</code>	54
5.13.2.10 <code>llc</code>	54
5.13.2.11 <code>snap</code>	54
5.13.2.12 <code>arp</code>	54
5.13.2.13 <code>ipv4</code>	55
5.13.2.14 <code>ipv6</code>	55
5.13.2.15 <code>frame</code>	55
5.14 <code>vtss_ace_vlan_t</code> Struct Reference	55
5.14.1 Detailed Description	55
5.14.2 Field Documentation	56
5.14.2.1 <code>vid</code>	56
5.14.2.2 <code>usr_prio</code>	56
5.14.2.3 <code>cfi</code>	56
5.14.2.4 <code>tagged</code>	56
5.15 <code>vtss_acl_action_t</code> Struct Reference	57
5.15.1 Detailed Description	57
5.15.2 Field Documentation	57
5.15.2.1 <code>cpu</code>	57
5.15.2.2 <code>cpu_once</code>	57
5.15.2.3 <code>cpu_queue</code>	58
5.15.2.4 <code>police</code>	58
5.15.2.5 <code>policer_no</code>	58
5.15.2.6 <code>learn</code>	58
5.15.2.7 <code>port_action</code>	58
5.15.2.8 <code>port_list</code>	59
5.15.2.9 <code>mirror</code>	59
5.15.2.10 <code>ptp_action</code>	59
5.15.2.11 <code>ptp</code>	59
5.16 <code>vtss_acl_policer_conf_t</code> Struct Reference	59
5.16.1 Detailed Description	60

---

5.16.2 Field Documentation . . . . .	60
5.16.2.1 bit_rate_enable . . . . .	60
5.16.2.2 bit_rate . . . . .	60
5.16.2.3 rate . . . . .	60
5.17 vtss_acl_port_conf_t Struct Reference . . . . .	61
5.17.1 Detailed Description . . . . .	61
5.17.2 Field Documentation . . . . .	61
5.17.2.1 policy_no . . . . .	61
5.17.2.2 action . . . . .	61
5.18 vtss_acl_ptp_action_conf_t Struct Reference . . . . .	61
5.18.1 Detailed Description . . . . .	62
5.18.2 Field Documentation . . . . .	62
5.18.2.1 response . . . . .	62
5.18.2.2 log_message_interval . . . . .	62
5.18.2.3 copy_smac_to_dmac . . . . .	62
5.18.2.4 set_smac_to_port_mac . . . . .	63
5.18.2.5 dom_sel . . . . .	63
5.19 vtss_aggr_mode_t Struct Reference . . . . .	63
5.19.1 Detailed Description . . . . .	63
5.19.2 Field Documentation . . . . .	63
5.19.2.1 smac_enable . . . . .	64
5.19.2.2 dmac_enable . . . . .	64
5.19.2.3 sip_dip_enable . . . . .	64
5.19.2.4 sport_dport_enable . . . . .	64
5.20 vtss_aneg_t Struct Reference . . . . .	64
5.20.1 Detailed Description . . . . .	65
5.20.2 Field Documentation . . . . .	65
5.20.2.1 obey_pause . . . . .	65
5.20.2.2 generate_pause . . . . .	65
5.21 vtss_api_lock_t Struct Reference . . . . .	65

---

5.21.1	Detailed Description	66
5.21.2	Field Documentation	66
5.21.2.1	inst	66
5.21.2.2	function	66
5.21.2.3	file	66
5.21.2.4	line	67
5.22	vtss_basic_counters_t Struct Reference	67
5.22.1	Detailed Description	67
5.22.2	Field Documentation	67
5.22.2.1	rx_frames	67
5.22.2.2	tx_frames	68
5.23	vtss_chip_id_t Struct Reference	68
5.23.1	Detailed Description	68
5.23.2	Field Documentation	68
5.23.2.1	part_number	68
5.23.2.2	revision	69
5.24	vtss_debug_info_t Struct Reference	69
5.24.1	Detailed Description	69
5.24.2	Field Documentation	69
5.24.2.1	layer	69
5.24.2.2	group	70
5.24.2.3	chip_no	70
5.24.2.4	port_list	70
5.24.2.5	full	70
5.24.2.6	clear	70
5.24.2.7	vml_format	71
5.25	vtss_debug_lock_t Struct Reference	71
5.25.1	Detailed Description	71
5.25.2	Field Documentation	71
5.25.2.1	chip_no	71

---

5.26 vtss_dgroup_port_conf_t Struct Reference . . . . .	72
5.26.1 Detailed Description . . . . .	72
5.26.2 Field Documentation . . . . .	72
5.26.2.1 dgroup_no . . . . .	72
5.27 vtss_dlb_policer_conf_t Struct Reference . . . . .	72
5.27.1 Detailed Description . . . . .	73
5.27.2 Field Documentation . . . . .	73
5.27.2.1 type . . . . .	73
5.27.2.2 enable . . . . .	73
5.27.2.3 cm . . . . .	73
5.27.2.4 cf . . . . .	73
5.27.2.5 line_rate . . . . .	74
5.27.2.6 cir . . . . .	74
5.27.2.7 cbs . . . . .	74
5.27.2.8 eir . . . . .	74
5.27.2.9 ebs . . . . .	74
5.28 vtss_eee_port_conf_t Struct Reference . . . . .	75
5.28.1 Detailed Description . . . . .	75
5.28.2 Field Documentation . . . . .	75
5.28.2.1 eee_ena . . . . .	75
5.28.2.2 eee_fast_queues . . . . .	75
5.28.2.3 tx_tw . . . . .	76
5.28.2.4 lp_advertisement . . . . .	76
5.28.2.5 optimized_for_power . . . . .	76
5.29 vtss_eee_port_counter_t Struct Reference . . . . .	76
5.29.1 Detailed Description . . . . .	77
5.29.2 Field Documentation . . . . .	77
5.29.2.1 fill_level_get . . . . .	77
5.29.2.2 fill_level_thres . . . . .	77
5.29.2.3 fill_level . . . . .	77

---

5.29.2.4 tx_out_bytes_get . . . . .	77
5.29.2.5 tx_out_bytes . . . . .	78
5.30 vtss_eee_port_state_t Struct Reference . . . . .	78
5.30.1 Detailed Description . . . . .	78
5.30.2 Field Documentation . . . . .	78
5.30.2.1 select . . . . .	78
5.30.2.2 val . . . . .	79
5.31 vtss_eps_port_conf_t Struct Reference . . . . .	79
5.31.1 Detailed Description . . . . .	79
5.31.2 Field Documentation . . . . .	79
5.31.2.1 type . . . . .	79
5.31.2.2 port_no . . . . .	80
5.32 vtss_ewis_aisl_cons_act_s Struct Reference . . . . .	80
5.32.1 Detailed Description . . . . .	80
5.32.2 Field Documentation . . . . .	80
5.32.2.1 ais_on_los . . . . .	80
5.32.2.2 ais_on_lof . . . . .	81
5.33 vtss_ewis_conf_s Struct Reference . . . . .	81
5.33.1 Detailed Description . . . . .	81
5.33.2 Field Documentation . . . . .	81
5.33.2.1 ewis_init_done . . . . .	82
5.33.2.2 static_conf . . . . .	82
5.33.2.3 ewis_mode . . . . .	82
5.33.2.4 section_cons_act . . . . .	82
5.33.2.5 section_txти . . . . .	82
5.33.2.6 force_mode . . . . .	83
5.33.2.7 path_txти . . . . .	83
5.33.2.8 tx_oh . . . . .	83
5.33.2.9 tx_oh_passthru . . . . .	83
5.33.2.10 exp_sl . . . . .	83

5.33.2.11 test_conf . . . . .	84
5.33.2.12 ewis_cntr_thresh_conf . . . . .	84
5.33.2.13 perf_mode . . . . .	84
5.34 vtss_ewis_cons_act_s Struct Reference . . . . .	84
5.34.1 Detailed Description . . . . .	84
5.34.2 Field Documentation . . . . .	85
5.34.2.1 aisl . . . . .	85
5.34.2.2 rdil . . . . .	85
5.34.2.3 fault . . . . .	85
5.35 vtss_ewis_counter_s Struct Reference . . . . .	85
5.35.1 Detailed Description . . . . .	86
5.35.2 Field Documentation . . . . .	86
5.35.2.1 pn_ebc_p . . . . .	86
5.35.2.2 pf_ebc_p . . . . .	86
5.35.2.3 pn_ebc_l . . . . .	86
5.35.2.4 pf_ebc_l . . . . .	86
5.35.2.5 pn_ebc_s . . . . .	87
5.36 vtss_ewis_counter_threshold_s Struct Reference . . . . .	87
5.36.1 Detailed Description . . . . .	87
5.36.2 Field Documentation . . . . .	87
5.36.2.1 n_ebc_thr_s . . . . .	87
5.36.2.2 n_ebc_thr_l . . . . .	88
5.36.2.3 f_ebc_thr_l . . . . .	88
5.36.2.4 n_ebc_thr_p . . . . .	88
5.36.2.5 f_ebc_thr_p . . . . .	88
5.37 vtss_ewis_defects_s Struct Reference . . . . .	88
5.37.1 Detailed Description . . . . .	89
5.37.2 Field Documentation . . . . .	89
5.37.2.1 dlos_s . . . . .	89
5.37.2.2 doof_s . . . . .	89

5.37.2.3	dl0f_s . . . . .	90
5.37.2.4	dais_l . . . . .	90
5.37.2.5	drdi_l . . . . .	90
5.37.2.6	dais_p . . . . .	90
5.37.2.7	dlop_p . . . . .	90
5.37.2.8	duneq_p . . . . .	91
5.37.2.9	drdi_p . . . . .	91
5.37.2.10	dlcd_p . . . . .	91
5.37.2.11	dplm_p . . . . .	91
5.37.2.12	dfais_p . . . . .	91
5.37.2.13	dfplm_p . . . . .	92
5.37.2.14	dfuneq_p . . . . .	92
5.38	vtss_ewis_fault_cons_act_s Struct Reference . . . . .	92
5.38.1	Detailed Description . . . . .	92
5.38.2	Field Documentation . . . . .	93
5.38.2.1	fault_on_feplmp . . . . .	93
5.38.2.2	fault_on_feaisp . . . . .	93
5.38.2.3	fault_on_rdil . . . . .	93
5.38.2.4	fault_on_sef . . . . .	93
5.38.2.5	fault_on_lof . . . . .	93
5.38.2.6	fault_on_los . . . . .	94
5.38.2.7	fault_on_aisl . . . . .	94
5.38.2.8	fault_on_lcdp . . . . .	94
5.38.2.9	fault_on_plmp . . . . .	94
5.38.2.10	fault_on_aisp . . . . .	94
5.38.2.11	fault_on_lopp . . . . .	95
5.39	vtss_ewis_force_mode_s Struct Reference . . . . .	95
5.39.1	Detailed Description . . . . .	95
5.39.2	Field Documentation . . . . .	95
5.39.2.1	line_rx_force . . . . .	95

5.39.2.2	line_tx_force	96
5.39.2.3	path_force	96
5.40	vtss_ewis_line_force_mode_s Struct Reference	96
5.40.1	Detailed Description	96
5.40.2	Field Documentation	96
5.40.2.1	force_ais	97
5.40.2.2	force_rdi	97
5.41	vtss_ewis_line_tx_force_mode_s Struct Reference	97
5.41.1	Detailed Description	97
5.41.2	Field Documentation	97
5.41.2.1	force_ais	98
5.41.2.2	force_rdi	98
5.42	vtss_ewis_path_force_mode_s Struct Reference	98
5.42.1	Detailed Description	98
5.42.2	Field Documentation	98
5.42.2.1	force_uneq	99
5.42.2.2	force_rdi	99
5.43	vtss_ewis_perf_mode_s Struct Reference	99
5.43.1	Detailed Description	99
5.43.2	Field Documentation	99
5.43.2.1	pn_ebc_mode_s	100
5.43.2.2	pn_ebc_mode_l	100
5.43.2.3	pf_ebc_mode_l	100
5.43.2.4	pn_ebc_mode_p	100
5.43.2.5	pf_ebc_mode_p	100
5.44	vtss_ewis_perf_s Struct Reference	101
5.44.1	Detailed Description	101
5.44.2	Field Documentation	101
5.44.2.1	pn_ebc_s	101
5.44.2.2	pn_ebc_l	101

5.44.2.3	pf_ebc_l	102
5.44.2.4	pn_ebc_p	102
5.44.2.5	pf_ebc_p	102
5.45	vtss_ewis_rdil_cons_act_s Struct Reference	102
5.45.1	Detailed Description	103
5.45.2	Field Documentation	103
5.45.2.1	rdil_on_los	103
5.45.2.2	rdil_on_lof	103
5.45.2.3	rdil_on_lopc	103
5.45.2.4	rdil_on_ais_l	103
5.46	vtss_ewis_sl_conf_s Struct Reference	104
5.46.1	Detailed Description	104
5.46.2	Field Documentation	104
5.46.2.1	exsl	104
5.47	vtss_ewis_static_conf_s Struct Reference	104
5.47.1	Detailed Description	105
5.47.2	Field Documentation	105
5.47.2.1	ewis_txctrl1	105
5.47.2.2	ewis_txctrl2	105
5.47.2.3	ewis_rx_ctrl1	106
5.47.2.4	ewis_mode_ctrl	106
5.47.2.5	ewis_tx_a1_a2	106
5.47.2.6	ewis_tx_c2_h1	106
5.47.2.7	ewis_tx_h2_h3	106
5.47.2.8	ewis_tx_z0_e1	107
5.47.2.9	ewis_rx_frm_ctrl1	107
5.47.2.10	ewis_rx_frm_ctrl2	107
5.47.2.11	ewis_lof_ctrl1	107
5.47.2.12	ewis_lof_ctrl2	107
5.47.2.13	ewis_rx_err_frc1	108

5.47.2.14 <code>ewis_pmtick_ctrl</code>	108
5.47.2.15 <code>ewis_cnt_cfg</code>	108
5.48 <code>vtss_ewis_status_s</code> Struct Reference	108
5.48.1 Detailed Description	108
5.48.2 Field Documentation	109
5.48.2.1 <code>fault</code>	109
5.48.2.2 <code>link_stat</code>	109
5.49 <code>vtss_ewis_test_conf_s</code> Struct Reference	109
5.49.1 Detailed Description	109
5.49.2 Field Documentation	110
5.49.2.1 <code>loopback</code>	110
5.49.2.2 <code>test_pattern_gen</code>	110
5.49.2.3 <code>test_pattern_ana</code>	110
5.50 <code>vtss_ewis_test_status_s</code> Struct Reference	110
5.50.1 Detailed Description	111
5.50.2 Field Documentation	111
5.50.2.1 <code>tstpat_cnt</code>	111
5.50.2.2 <code>ana_sync</code>	111
5.51 <code>vtss_ewis_tti_s</code> Struct Reference	111
5.51.1 Detailed Description	112
5.51.2 Field Documentation	112
5.51.2.1 <code>mode</code>	112
5.51.2.2 <code>tti</code>	112
5.51.2.3 <code>valid</code>	112
5.52 <code>vtss_ewis_tx_oh_s</code> Struct Reference	112
5.52.1 Detailed Description	113
5.52.2 Field Documentation	113
5.52.2.1 <code>tx_dcc_s</code>	113
5.52.2.2 <code>tx_e1</code>	113
5.52.2.3 <code>tx_f1</code>	114

5.52.2.4 tx_z0 . . . . .	114
5.52.2.5 tx_dcc_l . . . . .	114
5.52.2.6 tx_e2 . . . . .	114
5.52.2.7 tx_k1_k2 . . . . .	114
5.52.2.8 tx_s1 . . . . .	115
5.52.2.9 tx_z1_z2 . . . . .	115
5.52.2.10 tx_c2 . . . . .	115
5.52.2.11 tx_f2 . . . . .	115
5.52.2.12 tx_n1 . . . . .	115
5.52.2.13 tx_z3_z4 . . . . .	116
5.53 vtss_ewis_tx_passthru_s Struct Reference . . . . .	116
5.53.1 Detailed Description . . . . .	116
5.53.2 Field Documentation . . . . .	116
5.53.2.1 tx_j0 . . . . .	117
5.53.2.2 tx_z0 . . . . .	117
5.53.2.3 tx_b1 . . . . .	117
5.53.2.4 tx_e1 . . . . .	117
5.53.2.5 tx_f1 . . . . .	117
5.53.2.6 tx_dcc_s . . . . .	118
5.53.2.7 tx_soh . . . . .	118
5.53.2.8 tx_b2 . . . . .	118
5.53.2.9 tx_k1 . . . . .	118
5.53.2.10 tx_k2 . . . . .	118
5.53.2.11 tx_reil . . . . .	119
5.53.2.12 tx_dcc_l . . . . .	119
5.53.2.13 tx_s1 . . . . .	119
5.53.2.14 tx_e2 . . . . .	119
5.53.2.15 tx_z1_z2 . . . . .	119
5.53.2.16 tx_loh . . . . .	120
5.54 vtss_fan_conf_t Struct Reference . . . . .	120

---

5.54.1	Detailed Description	120
5.54.2	Field Documentation	120
5.54.2.1	fan_pwm_freq	120
5.54.2.2	fan_low_pol	121
5.54.2.3	fan_open_col	121
5.54.2.4	type	121
5.54.2.5	ppr	121
5.55	vtss_fdma_cfg_t Struct Reference	121
5.55.1	Detailed Description	122
5.55.2	Field Documentation	122
5.55.2.1	enable	122
5.55.2.2	rx_mtu	122
5.55.2.3	rx_buf_cnt	123
5.55.2.4	rx_alloc_cb	123
5.55.2.5	rx_dont_strip_vlan_tag	124
5.55.2.6	rx_dont_reinsert_vlan_tag	124
5.55.2.7	rx_allow_vlan_tag_mismatch	124
5.55.2.8	rx_allow_multiple_dcbs	125
5.55.2.9	rx_cb	125
5.55.2.10	tx_buf_cnt	125
5.55.2.11	tx_done_cb	126
5.56	vtss_fdma_ch_cfg_t Struct Reference	126
5.56.1	Detailed Description	127
5.56.2	Field Documentation	127
5.56.2.1	usage	127
5.56.2.2	xtr_grp	127
5.56.2.3	inj_grp_mask	128
5.56.2.4	list	128
5.56.2.5	xtr_cb	129
5.56.2.6	prio	129

5.56.2.7	chip_no	130
5.57	vtss_fdma_throttle_cfg_t Struct Reference	130
5.57.1	Detailed Description	130
5.57.2	Field Documentation	131
5.57.2.1	frm_limit_per_tick	131
5.57.2.2	byte_limit_per_tick	131
5.57.2.3	suspend_tick_cnt	131
5.58	vtss_fdma_tx_info_t Struct Reference	131
5.58.1	Detailed Description	132
5.58.2	Field Documentation	132
5.58.2.1	pre_cb_ctxt1	132
5.58.2.2	pre_cb_ctxt2	132
5.58.2.3	pre_cb	132
5.59	vtss_gpio_10g_gpio_mode_t Struct Reference	133
5.59.1	Detailed Description	133
5.59.2	Field Documentation	133
5.59.2.1	mode	133
5.59.2.2	port	133
5.59.2.3	input	134
5.59.2.4	in_sig	134
5.59.2.5	p_gpio	134
5.59.2.6	c_intrpt	134
5.59.2.7	source	134
5.59.2.8	aggr_intrpt	135
5.59.2.9	use_as_intrpt	135
5.59.2.10	p_gpio_intrpt	135
5.60	vtss_init_conf_t Struct Reference	135
5.60.1	Detailed Description	136
5.60.2	Field Documentation	136
5.60.2.1	reg_read	136

5.60.2.2	reg_write	136
5.60.2.3	miim_read	136
5.60.2.4	miim_write	136
5.60.2.5	mmd_read	137
5.60.2.6	mmd_read_inc	137
5.60.2.7	mmd_write	137
5.60.2.8	spi_read_write	137
5.60.2.9	spi_32bit_read_write	137
5.60.2.10	spi_64bit_read_write	138
5.60.2.11	warm_start_enable	138
5.60.2.12	restart_info_src	138
5.60.2.13	restart_info_port	138
5.60.2.14	mux_mode	138
5.60.2.15	pi	139
5.60.2.16	serdes	139
5.61	vtss_inst_create_t Struct Reference	139
5.61.1	Detailed Description	139
5.61.2	Field Documentation	139
5.61.2.1	target	140
5.62	vtss_intr_t Struct Reference	140
5.62.1	Detailed Description	140
5.62.2	Field Documentation	140
5.62.2.1	link_change	140
5.63	vtss_ip_addr_t Struct Reference	141
5.63.1	Detailed Description	141
5.63.2	Field Documentation	141
5.63.2.1	type	141
5.63.2.2	ipv4	141
5.63.2.3	ipv6	142
5.63.2.4	addr	142

5.64 vtss_ip_network_t Struct Reference . . . . .	142
5.64.1 Detailed Description . . . . .	142
5.64.2 Field Documentation . . . . .	142
5.64.2.1 address . . . . .	142
5.64.2.2 prefix_size . . . . .	143
5.65 vtss_ipv4_network_t Struct Reference . . . . .	143
5.65.1 Detailed Description . . . . .	143
5.65.2 Field Documentation . . . . .	143
5.65.2.1 address . . . . .	143
5.65.2.2 prefix_size . . . . .	144
5.66 vtss_ipv4_uc_t Struct Reference . . . . .	144
5.66.1 Detailed Description . . . . .	144
5.66.2 Field Documentation . . . . .	144
5.66.2.1 network . . . . .	144
5.66.2.2 destination . . . . .	145
5.67 vtss_ipv6_network_t Struct Reference . . . . .	145
5.67.1 Detailed Description . . . . .	145
5.67.2 Field Documentation . . . . .	145
5.67.2.1 address . . . . .	145
5.67.2.2 prefix_size . . . . .	146
5.68 vtss_ipv6_t Struct Reference . . . . .	146
5.68.1 Detailed Description . . . . .	146
5.68.2 Field Documentation . . . . .	146
5.68.2.1 addr . . . . .	146
5.69 vtss_ipv6_uc_t Struct Reference . . . . .	147
5.69.1 Detailed Description . . . . .	147
5.69.2 Field Documentation . . . . .	147
5.69.2.1 network . . . . .	147
5.69.2.2 destination . . . . .	147
5.70 vtss_irq_conf_t Struct Reference . . . . .	147

---

5.70.1	Detailed Description	148
5.70.2	Field Documentation	148
5.70.2.1	external	148
5.70.2.2	destination	148
5.71	vtss_irq_status_t Struct Reference	148
5.71.1	Detailed Description	149
5.71.2	Field Documentation	149
5.71.2.1	active	149
5.71.2.2	raw_ident	149
5.71.2.3	raw_status	149
5.71.2.4	raw_mask	150
5.72	vtss_l3_counters_t Struct Reference	150
5.72.1	Detailed Description	150
5.72.2	Field Documentation	150
5.72.2.1	ipv4uc_received_octets	150
5.72.2.2	ipv4uc_received_frames	151
5.72.2.3	ipv6uc_received_octets	151
5.72.2.4	ipv6uc_received_frames	151
5.72.2.5	ipv4uc_transmitted_octets	151
5.72.2.6	ipv4uc_transmitted_frames	151
5.72.2.7	ipv6uc_transmitted_octets	152
5.72.2.8	ipv6uc_transmitted_frames	152
5.73	vtss_lcpll_status_t Struct Reference	152
5.73.1	Detailed Description	152
5.73.2	Field Documentation	152
5.73.2.1	lock_status	153
5.73.2.2	cal_done	153
5.73.2.3	cal_error	153
5.73.2.4	fsm_lock	153
5.73.2.5	fsm_stat	153

5.73.2.6	gain_stat	154
5.74	vtss_learn_mode_t Struct Reference	154
5.74.1	Detailed Description	154
5.74.2	Field Documentation	154
5.74.2.1	automatic	154
5.74.2.2	cpu	155
5.74.2.3	discard	155
5.75	vtss_mac_t Struct Reference	155
5.75.1	Detailed Description	155
5.75.2	Field Documentation	155
5.75.2.1	addr	156
5.76	vtss_mac_table_entry_t Struct Reference	156
5.76.1	Detailed Description	156
5.76.2	Field Documentation	156
5.76.2.1	vid_mac	157
5.76.2.2	destination	157
5.76.2.3	copy_to_cpu	157
5.76.2.4	locked	157
5.76.2.5	aged	157
5.76.2.6	cpu_queue	158
5.76.2.7	enable	158
5.76.2.8	remote_entry	158
5.76.2.9	upsid	158
5.76.2.10	upspn	158
5.76.2.11	vstax2	159
5.77	vtss_mac_table_status_t Struct Reference	159
5.77.1	Detailed Description	159
5.77.2	Field Documentation	159
5.77.2.1	learned	159
5.77.2.2	replaced	160

5.77.2.3 moved . . . . .	160
5.77.2.4 aged . . . . .	160
5.78 vtss_mirror_conf_t Struct Reference . . . . .	160
5.78.1 Detailed Description . . . . .	161
5.78.2 Field Documentation . . . . .	161
5.78.2.1 port_no . . . . .	161
5.78.2.2 fwd_enable . . . . .	161
5.78.2.3 tag . . . . .	161
5.78.2.4 vid . . . . .	161
5.78.2.5 pcp . . . . .	162
5.78.2.6 dei . . . . .	162
5.79 vtss_mtimer_t Struct Reference . . . . .	162
5.79.1 Detailed Description . . . . .	162
5.79.2 Field Documentation . . . . .	162
5.79.2.1 timeout . . . . .	163
5.79.2.2 now . . . . .	163
5.80 vtss_npi_conf_t Struct Reference . . . . .	163
5.80.1 Detailed Description . . . . .	163
5.80.2 Field Documentation . . . . .	163
5.80.2.1 enable . . . . .	164
5.80.2.2 port_no . . . . .	164
5.81 vtss_os_timestamp_t Struct Reference . . . . .	164
5.81.1 Detailed Description . . . . .	164
5.81.2 Field Documentation . . . . .	164
5.81.2.1 hw_cnt . . . . .	165
5.82 vtss_packet_dma_conf_t Struct Reference . . . . .	165
5.82.1 Detailed Description . . . . .	165
5.82.2 Field Documentation . . . . .	165
5.82.2.1 dma_enable . . . . .	165
5.83 vtss_packet_frame_info_t Struct Reference . . . . .	166

---

---

5.83.1	Detailed Description	166
5.83.2	Field Documentation	166
5.83.2.1	port_no	166
5.83.2.2	glag_no	166
5.83.2.3	vid	167
5.83.2.4	port_tx	167
5.83.2.5	aggr_rx_disable	167
5.83.2.6	aggr_tx_disable	167
5.84	vtss_packet_port_filter_t Struct Reference	167
5.84.1	Detailed Description	168
5.84.2	Field Documentation	168
5.84.2.1	filter	168
5.84.2.2	tpid	168
5.85	vtss_packet_port_info_t Struct Reference	168
5.85.1	Detailed Description	169
5.85.2	Field Documentation	169
5.85.2.1	port_no	169
5.85.2.2	glag_no	169
5.85.2.3	vid	169
5.85.2.4	aggr_rx_disable	170
5.85.2.5	aggr_tx_disable	170
5.86	vtss_packet_rx_conf_t Struct Reference	170
5.86.1	Detailed Description	170
5.86.2	Field Documentation	170
5.86.2.1	queue	171
5.86.2.2	reg	171
5.86.2.3	map	171
5.86.2.4	grp_map	171
5.87	vtss_packet_rx_header_t Struct Reference	171
5.87.1	Detailed Description	172

---

5.87.2 Field Documentation . . . . .	172
5.87.2.1 length . . . . .	172
5.87.2.2 port_no . . . . .	172
5.87.2.3 queue_mask . . . . .	172
5.87.2.4 learn . . . . .	173
5.87.2.5 arrived_tagged . . . . .	173
5.87.2.6 tag . . . . .	173
5.87.2.7 vstax . . . . .	173
5.88 vtss_packet_rx_info_t Struct Reference . . . . .	173
5.88.1 Detailed Description . . . . .	174
5.88.2 Field Documentation . . . . .	174
5.88.2.1 hints . . . . .	175
5.88.2.2 length . . . . .	175
5.88.2.3 port_no . . . . .	175
5.88.2.4 glag_no . . . . .	176
5.88.2.5 tag_type . . . . .	176
5.88.2.6 tag . . . . .	177
5.88.2.7 stripped_tag . . . . .	177
5.88.2.8 xtr_qu_mask . . . . .	178
5.88.2.9 cos . . . . .	178
5.88.2.10 acl_hit . . . . .	178
5.88.2.11 acl_idx . . . . .	179
5.88.2.12 sw_tstamp . . . . .	179
5.88.2.13 tstamp_id . . . . .	179
5.88.2.14 tstamp_id_decoded . . . . .	180
5.88.2.15 hw_tstamp . . . . .	180
5.88.2.16 hw_tstamp_decoded . . . . .	180
5.88.2.17 sflow_type . . . . .	181
5.88.2.18 sflow_port_no . . . . .	181
5.88.2.19 oam_info . . . . .	182

5.88.2.20 oam_info_decoded . . . . .	182
5.88.2.21 isdx . . . . .	182
5.88.2.22 vstax . . . . .	183
5.89 vtss_packet_rx_meta_t Struct Reference . . . . .	183
5.89.1 Detailed Description . . . . .	183
5.89.2 Field Documentation . . . . .	184
5.89.2.1 no_wait . . . . .	184
5.89.2.2 chip_no . . . . .	184
5.89.2.3 xtr_qu . . . . .	185
5.89.2.4 etype . . . . .	185
5.89.2.5 fcs . . . . .	186
5.89.2.6 sw_tstamp . . . . .	186
5.89.2.7 length . . . . .	187
5.90 vtss_packet_rx_port_conf_t Struct Reference . . . . .	187
5.90.1 Detailed Description . . . . .	187
5.90.2 Field Documentation . . . . .	188
5.90.2.1 bpdu_reg . . . . .	188
5.90.2.2 garp_reg . . . . .	188
5.91 vtss_packet_rx_queue_conf_t Struct Reference . . . . .	188
5.91.1 Detailed Description . . . . .	188
5.91.2 Field Documentation . . . . .	188
5.91.2.1 size . . . . .	189
5.91.2.2 npi . . . . .	189
5.92 vtss_packet_rx_queue_map_t Struct Reference . . . . .	189
5.92.1 Detailed Description . . . . .	189
5.92.2 Field Documentation . . . . .	190
5.92.2.1 bpdu_queue . . . . .	190
5.92.2.2 garp_queue . . . . .	190
5.92.2.3 learn_queue . . . . .	190
5.92.2.4 igmp_queue . . . . .	190

5.92.2.5  ipmc_ctrl_queue . . . . .	190
5.92.2.6  mac_vid_queue . . . . .	191
5.92.2.7  stack_queue . . . . .	191
5.92.2.8  sflow_queue . . . . .	191
5.92.2.9  lrn_all_queue . . . . .	191
5.93  vtss_packet_rx_queue_npi_conf_t Struct Reference . . . . .	191
5.93.1  Detailed Description . . . . .	192
5.93.2  Field Documentation . . . . .	192
5.93.2.1  enable . . . . .	192
5.94  vtss_packet_rx_reg_t Struct Reference . . . . .	192
5.94.1  Detailed Description . . . . .	192
5.94.2  Field Documentation . . . . .	193
5.94.2.1  bpdu_cpu_only . . . . .	193
5.94.2.2  garp_cpu_only . . . . .	193
5.94.2.3  ipmc_ctrl_cpu_copy . . . . .	193
5.94.2.4  igmp_cpu_only . . . . .	193
5.94.2.5  mld_cpu_only . . . . .	194
5.95  vtss_packet_tx_ifh_t Struct Reference . . . . .	194
5.95.1  Detailed Description . . . . .	194
5.95.2  Field Documentation . . . . .	194
5.95.2.1  length . . . . .	194
5.95.2.2  ifh . . . . .	195
5.96  vtss_packet_tx_info_t Struct Reference . . . . .	195
5.96.1  Detailed Description . . . . .	195
5.96.2  Field Documentation . . . . .	196
5.96.2.1  switch_frm . . . . .	196
5.96.2.2  dst_port_mask . . . . .	196
5.96.2.3  frm_len . . . . .	197
5.96.2.4  tag . . . . .	197
5.96.2.5  aggr_code . . . . .	198

5.96.2.6 cos . . . . .	199
5.96.2.7 tx_vstax_hdr . . . . .	199
5.96.2.8 bin . . . . .	200
5.96.2.9 sym . . . . .	200
5.96.2.10 vstax . . . . .	201
5.96.2.11 ptp_action . . . . .	201
5.96.2.12 ptp_id . . . . .	201
5.96.2.13 ptp_timestamp . . . . .	202
5.96.2.14 latch_timestamp . . . . .	202
5.96.2.15 oam_type . . . . .	203
5.96.2.16 isdx . . . . .	203
5.96.2.17 isdx_dont_use . . . . .	204
5.96.2.18 dp . . . . .	204
5.96.2.19 masquerade_port . . . . .	205
5.96.2.20 pdu_offset . . . . .	205
5.97 vtss_phy_10g_apc_conf_t Struct Reference . . . . .	206
5.97.1 Detailed Description . . . . .	206
5.97.2 Field Documentation . . . . .	206
5.97.2.1 op_mode . . . . .	206
5.97.2.2 op_mode_flag . . . . .	207
5.98 vtss_phy_10g_apc_status_t Struct Reference . . . . .	207
5.98.1 Detailed Description . . . . .	207
5.98.2 Field Documentation . . . . .	207
5.98.2.1 reset . . . . .	207
5.98.2.2 freeze . . . . .	208
5.99 vtss_phy_10g_auto_failover_conf_t Struct Reference . . . . .	208
5.99.1 Detailed Description . . . . .	208
5.99.2 Field Documentation . . . . .	208
5.99.2.1 port_no . . . . .	208
5.99.2.2 evnt . . . . .	209

5.99.2.3 trig_ch_id . . . . .	209
5.99.2.4 is_host_side . . . . .	209
5.99.2.5 channel_id . . . . .	209
5.99.2.6 v_gpio . . . . .	209
5.99.2.7 a_gpio . . . . .	210
5.99.2.8 enable . . . . .	210
5.99.2.9 filter . . . . .	210
5.99.2.10 fltr_val . . . . .	210
5.100vtss_phy_10g_base_kr_autoneg_t Struct Reference . . . . .	210
5.100.1 Detailed Description . . . . .	211
5.100.2 Field Documentation . . . . .	211
5.100.2.1 an_restart . . . . .	211
5.100.2.2 an_enable . . . . .	211
5.100.2.3 an_reset . . . . .	211
5.101vtss_phy_10g_base_kr_conf_t Struct Reference . . . . .	212
5.101.1 Detailed Description . . . . .	212
5.101.2 Field Documentation . . . . .	212
5.101.2.1 cm1 . . . . .	212
5.101.2.2 c0 . . . . .	212
5.101.2.3 c1 . . . . .	213
5.101.2.4 ampl . . . . .	213
5.101.2.5 slewrate . . . . .	213
5.101.2.6 en_ob . . . . .	213
5.101.2.7 ser_inv . . . . .	213
5.102vtss_phy_10g_base_kr_ld_adv_abil_t Struct Reference . . . . .	214
5.102.1 Detailed Description . . . . .	214
5.102.2 Field Documentation . . . . .	214
5.102.2.1 adv_1g . . . . .	214
5.102.2.2 adv_10g . . . . .	214
5.102.2.3 fec_abil . . . . .	215

5.102.2.4 fec_req . . . . .	215
5.103vtss_phy_10g_base_kr_status_t Struct Reference . . . . .	215
5.103.1 Detailed Description . . . . .	215
5.103.2 Field Documentation . . . . .	215
5.103.2.1 aneg . . . . .	216
5.103.2.2 train . . . . .	216
5.103.2.3 fec . . . . .	216
5.104vtss_phy_10g_base_kr_train_aneg_t Struct Reference . . . . .	216
5.104.1 Detailed Description . . . . .	217
5.104.2 Field Documentation . . . . .	217
5.104.2.1 training . . . . .	217
5.104.2.2 autoneg . . . . .	217
5.104.2.3 ld_abil . . . . .	217
5.104.2.4 host_kr . . . . .	217
5.104.2.5 line_kr . . . . .	218
5.105vtss_phy_10g_base_kr_training_t Struct Reference . . . . .	218
5.105.1 Detailed Description . . . . .	218
5.105.2 Field Documentation . . . . .	218
5.105.2.1 enable . . . . .	218
5.105.2.2 trmthd_cp . . . . .	219
5.105.2.3 trmthd_c0 . . . . .	219
5.105.2.4 trmthd_cm . . . . .	219
5.105.2.5 ld_pre_init . . . . .	219
5.106vtss_phy_10g_ckout_conf_t Struct Reference . . . . .	219
5.106.1 Detailed Description . . . . .	220
5.106.2 Field Documentation . . . . .	220
5.106.2.1 mode . . . . .	220
5.106.2.2 src . . . . .	220
5.106.2.3 freq . . . . .	220
5.106.2.4 squelch_inv . . . . .	221

5.106.2.5 enable . . . . .	221
5.106.2.6 ckout_sel . . . . .	221
5.107vtss_phy_10g_clause_37_adv_t Struct Reference . . . . .	221
5.107.1 Detailed Description . . . . .	222
5.107.2 Field Documentation . . . . .	222
5.107.2.1 fdx . . . . .	222
5.107.2.2 hdx . . . . .	222
5.107.2.3 symmetric_pause . . . . .	222
5.107.2.4 asymmetric_pause . . . . .	222
5.107.2.5 remote_fault . . . . .	223
5.107.2.6 acknowledge . . . . .	223
5.107.2.7 next_page . . . . .	223
5.108vtss_phy_10g_clause_37_cmn_status_t Struct Reference . . . . .	223
5.108.1 Detailed Description . . . . .	223
5.108.2 Field Documentation . . . . .	224
5.108.2.1 line . . . . .	224
5.108.2.2 host . . . . .	224
5.109vtss_phy_10g_clause_37_control_t Struct Reference . . . . .	224
5.109.1 Detailed Description . . . . .	224
5.109.2 Field Documentation . . . . .	225
5.109.2.1 enable . . . . .	225
5.109.2.2 advertisement . . . . .	225
5.109.2.3 enable_pass_thru . . . . .	225
5.109.2.4 line . . . . .	225
5.109.2.5 host . . . . .	225
5.109.2.6 l_h . . . . .	226
5.110vtss_phy_10g_clause_37_status_t Struct Reference . . . . .	226
5.110.1 Detailed Description . . . . .	226
5.110.2 Field Documentation . . . . .	226
5.110.2.1 link . . . . .	226

5.110.2.2 complete . . . . .	227
5.110.2.3 partner_advertisement . . . . .	227
5.110.2.4 autoneg . . . . .	227
5.111vtss_phy_10g_clk_src_t Struct Reference . . . . .	227
5.111.1 Detailed Description . . . . .	227
5.111.2 Field Documentation . . . . .	228
5.111.2.1 is_high_amp . . . . .	228
5.112vtss_phy_10g_cnt_t Struct Reference . . . . .	228
5.112.1 Detailed Description . . . . .	228
5.112.2 Field Documentation . . . . .	228
5.112.2.1 pcs . . . . .	228
5.113vtss_phy_10g_fifo_sync_t Struct Reference . . . . .	229
5.113.1 Detailed Description . . . . .	229
5.113.2 Field Documentation . . . . .	229
5.113.2.1 bypass_in_api . . . . .	229
5.113.2.2 skip_rev_check . . . . .	229
5.113.2.3 pr . . . . .	230
5.114vtss_phy_10g_fw_status_t Struct Reference . . . . .	230
5.114.1 Detailed Description . . . . .	230
5.114.2 Field Documentation . . . . .	230
5.114.2.1 edc_fw_rev . . . . .	230
5.114.2.2 edc_fw_chksum . . . . .	231
5.114.2.3 icpu_activity . . . . .	231
5.114.2.4 edc_fw_api_load . . . . .	231
5.115vtss_phy_10g_host_clk_conf_t Struct Reference . . . . .	231
5.115.1 Detailed Description . . . . .	232
5.115.2 Field Documentation . . . . .	232
5.115.2.1 mode . . . . .	232
5.115.2.2 recvrd_clk_sel . . . . .	232
5.115.2.3 clk_sel_no . . . . .	232

5.116vtss_phy_10g_ib_conf_t Struct Reference . . . . .	232
5.116.1 Detailed Description . . . . .	233
5.116.2 Field Documentation . . . . .	233
5.116.2.1 offs . . . . .	233
5.116.2.2 gain . . . . .	233
5.116.2.3 gainadj . . . . .	234
5.116.2.4 l . . . . .	234
5.116.2.5 c . . . . .	234
5.116.2.6 agc . . . . .	234
5.116.2.7 dfe1 . . . . .	234
5.116.2.8 dfe2 . . . . .	235
5.116.2.9 dfe3 . . . . .	235
5.116.2.10dfe4 . . . . .	235
5.116.2.11ld . . . . .	235
5.116.2.12prbs . . . . .	235
5.116.2.13prbs_inv . . . . .	236
5.116.2.14apc_bit_mask . . . . .	236
5.116.2.15freeze_bit_mask . . . . .	236
5.116.2.16config_bit_mask . . . . .	236
5.116.2.17s_host . . . . .	236
5.117vtss_phy_10g_ib_status_t Struct Reference . . . . .	237
5.117.1 Detailed Description . . . . .	237
5.117.2 Field Documentation . . . . .	237
5.117.2.1 ib_conf . . . . .	237
5.117.2.2 sig_det . . . . .	237
5.117.2.3 bit_errors . . . . .	238
5.118vtss_phy_10g_ib_storage_t Struct Reference . . . . .	238
5.118.1 Detailed Description . . . . .	238
5.118.2 Field Documentation . . . . .	238
5.118.2.1 ib_storage_bool . . . . .	238

5.118.2.2 <code>ib_storage</code>	239
5.119 <code>vtss_phy_10g_id_t</code> Struct Reference	239
5.119.1 Detailed Description	239
5.119.2 Field Documentation	239
5.119.2.1 <code>part_number</code>	239
5.119.2.2 <code>revision</code>	240
5.119.2.3 <code>channel_id</code>	240
5.119.2.4 <code>family</code>	240
5.119.2.5 <code>type</code>	240
5.119.2.6 <code>phy_api_base_no</code>	240
5.119.2.7 <code>device_feature_status</code>	241
5.120 <code>vtss_phy_10g_init_parm_t</code> Struct Reference	241
5.120.1 Detailed Description	241
5.120.2 Field Documentation	241
5.120.2.1 <code>channel_conf</code>	241
5.121 <code>vtss_phy_10g_jitter_conf_t</code> Struct Reference	242
5.121.1 Detailed Description	242
5.121.2 Field Documentation	242
5.121.2.1 <code>incr_levn</code>	242
5.121.2.2 <code>levn</code>	242
5.121.2.3 <code>vtail</code>	243
5.122 <code>vtss_phy_10g_kr_status_aneg_t</code> Struct Reference	243
5.122.1 Detailed Description	243
5.122.2 Field Documentation	243
5.122.2.1 <code>complete</code>	243
5.122.2.2 <code>active</code>	244
5.122.2.3 <code>request_10g</code>	244
5.122.2.4 <code>request_1g</code>	244
5.122.2.5 <code>request_fec_change</code>	244
5.122.2.6 <code>fec_enable</code>	244

5.122.2.7 sm . . . . .	245
5.122.2.8 lp_aneg_able . . . . .	245
5.122.2.9 block_lock . . . . .	245
5.123vtss_phy_10g_kr_status_fec_t Struct Reference . . . . .	245
5.123.1 Detailed Description . . . . .	245
5.123.2 Field Documentation . . . . .	246
5.123.2.1 enable . . . . .	246
5.123.2.2 corrected_block_cnt . . . . .	246
5.123.2.3 uncorrected_block_cnt . . . . .	246
5.124vtss_phy_10g_kr_status_train_t Struct Reference . . . . .	246
5.124.1 Detailed Description . . . . .	247
5.124.2 Field Documentation . . . . .	247
5.124.2.1 complete . . . . .	247
5.124.2.2 cm_ob_tap_result . . . . .	247
5.124.2.3 cp_ob_tap_result . . . . .	247
5.124.2.4 c0_ob_tap_result . . . . .	247
5.125vtss_phy_10g_lane_sync_conf_t Struct Reference . . . . .	248
5.125.1 Detailed Description . . . . .	248
5.125.2 Field Documentation . . . . .	248
5.125.2.1 enable . . . . .	248
5.125.2.2 tx_macro . . . . .	248
5.125.2.3 rx_macro . . . . .	249
5.125.2.4 rx_ch . . . . .	249
5.125.2.5 tx_ch . . . . .	249
5.126vtss_phy_10g_line_clk_conf_t Struct Reference . . . . .	249
5.126.1 Detailed Description . . . . .	250
5.126.2 Field Documentation . . . . .	250
5.126.2.1 mode . . . . .	250
5.126.2.2 recvrd_clk_sel . . . . .	250
5.126.2.3 clk_sel_no . . . . .	250

---

5.127vtss_phy_10g_loopback_t Struct Reference . . . . .	250
5.127.1 Detailed Description . . . . .	251
5.127.2 Field Documentation . . . . .	251
5.127.2.1 lb_type . . . . .	251
5.127.2.2 enable . . . . .	251
5.128vtss_phy_10g_mode_t Struct Reference . . . . .	251
5.128.1 Detailed Description . . . . .	253
5.128.2 Member Enumeration Documentation . . . . .	253
5.128.2.1 anonymous enum . . . . .	253
5.128.3 Field Documentation . . . . .	253
5.128.3.1 oper_mode . . . . .	253
5.128.3.2 interface . . . . .	253
5.128.3.3 wrefclk . . . . .	254
5.128.3.4 high_input_gain . . . . .	254
5.128.3.5 xfi_pol_invert . . . . .	254
5.128.3.6 xauilane_flip . . . . .	254
5.128.3.7 channel_id . . . . .	254
5.128.3.8 hl_clk_synth . . . . .	255
5.128.3.9 rcvrd_clk . . . . .	255
5.128.3.10rcvrd_clk_div . . . . .	255
5.128.3.11sref_clk_div . . . . .	255
5.128.3.12wref_clk_div . . . . .	255
5.128.3.13edc_fw_load . . . . .	256
5.128.3.14use_conf . . . . .	256
5.128.3.15ob_conf . . . . .	256
5.128.3.16b_conf . . . . .	256
5.128.3.17dig_offset_reg . . . . .	256
5.128.3.18apc_offs_ctrl . . . . .	257
5.128.3.19apc_line_id_ctrl . . . . .	257
5.128.3.20apc_host_id_ctrl . . . . .	257

5.128.3.21d_filter . . . . .	257
5.128.3.22cfg0 . . . . .	257
5.128.3.23b_ini_lp . . . . .	258
5.128.3.24b_min_lp . . . . .	258
5.128.3.25b_max_lp . . . . .	258
5.128.3.26apc_eqz_offs_par_cfg . . . . .	258
5.128.3.27apc_line_eqz_Id_ctrl . . . . .	258
5.128.3.28apc_host_eqz_Id_ctrl . . . . .	259
5.128.3.29_offset_guard . . . . .	259
5.128.3.30h_offset_guard . . . . .	259
5.128.3.31serdes_conf . . . . .	259
5.128.3.32apc_ib_regulator . . . . .	259
5.128.3.33pma_txratecontrol . . . . .	260
5.128.3.34venice_rev_a_los_detection_workaround . . . . .	260
5.128.3.35ddr_mode . . . . .	260
5.128.3.36master . . . . .	260
5.128.3.37rate . . . . .	260
5.128.3.38polarity . . . . .	261
5.128.3.39s_host_wan . . . . .	261
5.128.3.40h_clk_src . . . . .	261
5.128.3.41l_clk_src . . . . .	261
5.128.3.42ref_for_host . . . . .	261
5.128.3.43ink_6g_distance . . . . .	262
5.128.3.44h_media . . . . .	262
5.128.3.45_media . . . . .	262
5.128.3.46h_ib_conf . . . . .	262
5.128.3.47_ib_conf . . . . .	262
5.128.3.48h_apc_conf . . . . .	263
5.128.3.49_apc_conf . . . . .	263
5.128.3.50enable_pass_thru . . . . .	263

5.128.3.5 <code>is_init</code>	263
5.128.3.52 <code>sd6g_calib_done</code>	263
5.129 <code>vtss_phy_10g_ob_status_t</code> Struct Reference	264
5.129.1 Detailed Description	264
5.129.2 Field Documentation	264
5.129.2.1 <code>r_ctrl</code>	264
5.129.2.2 <code>c_ctrl</code>	264
5.129.2.3 <code>slew</code>	265
5.129.2.4 <code>levn</code>	265
5.129.2.5 <code>d_fltr</code>	265
5.129.2.6 <code>v3</code>	265
5.129.2.7 <code>vp</code>	265
5.129.2.8 <code>v4</code>	266
5.129.2.9 <code>v5</code>	266
5.129.2.10 <code>s_host</code>	266
5.130 <code>vtss_phy_10g_pcs_prbs_gen_conf_t</code> Struct Reference	266
5.130.1 Detailed Description	266
5.130.2 Field Documentation	267
5.130.2.1 <code>prbs_gen</code>	267
5.131 <code>vtss_phy_10g_pcs_prbs_mon_conf_t</code> Struct Reference	267
5.131.1 Detailed Description	267
5.131.2 Field Documentation	267
5.131.2.1 <code>prbs_mon</code>	267
5.131.2.2 <code>error_counter</code>	268
5.132 <code>vtss_phy_10g_pkt_gen_conf_t</code> Struct Reference	268
5.132.1 Detailed Description	268
5.132.2 Field Documentation	268
5.132.2.1 <code>enable</code>	269
5.132.2.2 <code>ptp</code>	269
5.132.2.3 <code>ingress</code>	269

5.132.2.4 frames . . . . .	269
5.132.2.5 frame_single . . . . .	269
5.132.2.6 etype . . . . .	270
5.132.2.7 pkt_len . . . . .	270
5.132.2.8 ipg_len . . . . .	270
5.132.2.9 smac . . . . .	270
5.132.2.10dmac . . . . .	270
5.132.2.11ptp_ts_sec . . . . .	271
5.132.2.12ptp_ts_ns . . . . .	271
5.132.2.13srate . . . . .	271
5.133vtss_phy_10g_pkt_mon_conf_t Struct Reference . . . . .	271
5.133.1 Detailed Description . . . . .	272
5.133.2 Field Documentation . . . . .	272
5.133.2.1 enable . . . . .	272
5.133.2.2 update . . . . .	272
5.133.2.3 reset . . . . .	272
5.133.2.4 good_crc . . . . .	272
5.133.2.5 bad_crc . . . . .	273
5.133.2.6 frag . . . . .	273
5.133.2.7 Ifault . . . . .	273
5.133.2.8 ber . . . . .	273
5.134vtss_phy_10g_polarity_inv_t Struct Reference . . . . .	273
5.134.1 Detailed Description . . . . .	274
5.134.2 Field Documentation . . . . .	274
5.134.2.1 line_rx . . . . .	274
5.134.2.2 line_tx . . . . .	274
5.134.2.3 host_rx . . . . .	274
5.134.2.4 host_tx . . . . .	275
5.135vtss_phy_10g_prbs_gen_conf_t Struct Reference . . . . .	275
5.135.1 Detailed Description . . . . .	275

---

5.135.2 Field Documentation . . . . .	275
5.135.2.1 enable . . . . .	275
5.135.2.2 prbsn_tx_sel . . . . .	276
5.135.2.3 line . . . . .	276
5.135.2.4 prbsn_tx_io . . . . .	276
5.135.2.5 prbsn_tx_iw . . . . .	276
5.136vtss_phy_10g_prbs_mon_conf_t Struct Reference . . . . .	276
5.136.1 Detailed Description . . . . .	277
5.136.2 Field Documentation . . . . .	277
5.136.2.1 enable . . . . .	277
5.136.2.2 line . . . . .	277
5.136.2.3 max_bist_frames . . . . .	278
5.136.2.4 error_states . . . . .	278
5.136.2.5 des_interface_width . . . . .	278
5.136.2.6 prbsn_sel . . . . .	278
5.136.2.7 prbs_check_input_invert . . . . .	278
5.136.2.8 no_of_errors . . . . .	279
5.136.2.9 bist_mode . . . . .	279
5.136.2.10error_status . . . . .	279
5.136.2.11PRBS_status . . . . .	279
5.136.2.12main_status . . . . .	279
5.136.2.13stuck_at_par . . . . .	280
5.136.2.14stuck_at_01 . . . . .	280
5.136.2.15no_sync . . . . .	280
5.136.2.16nstable . . . . .	280
5.136.2.17incomplete . . . . .	280
5.136.2.18active . . . . .	281
5.137vtss_phy_10g_rxckout_conf_t Struct Reference . . . . .	281
5.137.1 Detailed Description . . . . .	281
5.137.2 Field Documentation . . . . .	281

5.137.2.1 mode . . . . .	281
5.137.2.2 squelch_on_pcs_fault . . . . .	282
5.137.2.3 squelch_on_lopc . . . . .	282
5.138vtss_phy_10g_sckout_conf_t Struct Reference . . . . .	282
5.138.1 Detailed Description . . . . .	282
5.138.2 Field Documentation . . . . .	282
5.138.2.1 mode . . . . .	283
5.138.2.2 src . . . . .	283
5.138.2.3 freq . . . . .	283
5.138.2.4 squelch_inv . . . . .	283
5.138.2.5 enable . . . . .	283
5.139vtss_phy_10g_serdes_status_t Struct Reference . . . . .	284
5.139.1 Detailed Description . . . . .	284
5.139.2 Field Documentation . . . . .	284
5.139.2.1 rcomp . . . . .	285
5.139.2.2 h_pll5g_lock_status . . . . .	285
5.139.2.3 h_pll5g_fsm_lock . . . . .	285
5.139.2.4 h_pll5g_fsm_stat . . . . .	285
5.139.2.5 h_pll5g_gain . . . . .	285
5.139.2.6 l_pll5g_lock_status . . . . .	286
5.139.2.7 l_pll5g_fsm_lock . . . . .	286
5.139.2.8 l_pll5g_fsm_stat . . . . .	286
5.139.2.9 l_pll5g_gain . . . . .	286
5.139.2.10h_rx_rcpll_lock_status . . . . .	286
5.139.2.11h_rx_rcpll_range . . . . .	287
5.139.2.12h_rx_rcpll_vco_load . . . . .	287
5.139.2.13h_rx_rcpll_fsm_status . . . . .	287
5.139.2.14_rx_rcpll_lock_status . . . . .	287
5.139.2.15_rx_rcpll_range . . . . .	287
5.139.2.16_rx_rcpll_vco_load . . . . .	288

5.139.2.17_rx_rcpll_fsm_status . . . . .	288
5.139.2.18_tx_rcpll_lock_status . . . . .	288
5.139.2.19_tx_rcpll_range . . . . .	288
5.139.2.20_tx_rcpll_vco_load . . . . .	288
5.139.2.21_tx_rcpll_fsm_status . . . . .	289
5.139.2.22_tx_rcpll_lock_status . . . . .	289
5.139.2.23_tx_rcpll_range . . . . .	289
5.139.2.24_tx_rcpll_vco_load . . . . .	289
5.139.2.25_tx_rcpll_fsm_status . . . . .	289
5.139.2.26_pma . . . . .	290
5.139.2.27_h_pcs . . . . .	290
5.139.2.28_pma . . . . .	290
5.139.2.29_pcs . . . . .	290
5.139.2.30_wis . . . . .	290
5.140_vtss_phy_10g_srefclk_mode_t Struct Reference . . . . .	291
5.140.1 Detailed Description . . . . .	291
5.140.2 Field Documentation . . . . .	291
5.140.2.1 enable . . . . .	291
5.140.2.2 freq . . . . .	291
5.141_vtss_phy_10g_status_t Struct Reference . . . . .	291
5.141.1 Detailed Description . . . . .	292
5.141.2 Field Documentation . . . . .	292
5.141.2.1 pma . . . . .	292
5.141.2.2 hpma . . . . .	292
5.141.2.3 wis . . . . .	292
5.141.2.4 pcs . . . . .	293
5.141.2.5 hpcs . . . . .	293
5.141.2.6 xs . . . . .	293
5.141.2.7 lpcs_1g . . . . .	293
5.141.2.8 hpcs_1g . . . . .	293

5.141.2.9 status . . . . .	294
5.141.2.10 block_lock . . . . .	294
5.141.2.11 lopc_stat . . . . .	294
5.142 vtss_phy_10g_timestamp_val_t Struct Reference . . . . .	294
5.142.1 Detailed Description . . . . .	294
5.142.2 Field Documentation . . . . .	295
5.142.2.1 timestamp . . . . .	295
5.143 vtss_phy_10g_txckout_conf_t Struct Reference . . . . .	295
5.143.1 Detailed Description . . . . .	295
5.143.2 Field Documentation . . . . .	295
5.143.2.1 mode . . . . .	295
5.144 vtss_phy_10g_vscope_conf_t Struct Reference . . . . .	296
5.144.1 Detailed Description . . . . .	296
5.144.2 Field Documentation . . . . .	296
5.144.2.1 scan_type . . . . .	296
5.144.2.2 line . . . . .	296
5.144.2.3 enable . . . . .	296
5.144.2.4 error_thres . . . . .	297
5.145 vtss_phy_10g_vscope_scan_conf_t Struct Reference . . . . .	297
5.145.1 Detailed Description . . . . .	297
5.145.2 Field Documentation . . . . .	297
5.145.2.1 line . . . . .	297
5.145.2.2 x_start . . . . .	298
5.145.2.3 y_start . . . . .	298
5.145.2.4 x_incr . . . . .	298
5.145.2.5 y_incr . . . . .	298
5.145.2.6 x_count . . . . .	298
5.145.2.7 y_count . . . . .	299
5.145.2.8 ber . . . . .	299
5.146 vtss_phy_10g_vscope_scan_status_t Struct Reference . . . . .	299

5.146.1 Detailed Description . . . . .	299
5.146.2 Field Documentation . . . . .	299
5.146.2.1 scan_conf . . . . .	300
5.146.2.2 error_free_x . . . . .	300
5.146.2.3 error_free_y . . . . .	300
5.146.2.4 amp_range . . . . .	300
5.146.2.5 errors . . . . .	300
5.147vtss_phy_aneg_t Struct Reference . . . . .	301
5.147.1 Detailed Description . . . . .	301
5.147.2 Field Documentation . . . . .	301
5.147.2.1 speed_10m_hdx . . . . .	301
5.147.2.2 speed_10m_fdx . . . . .	301
5.147.2.3 speed_100m_hdx . . . . .	302
5.147.2.4 speed_100m_fdx . . . . .	302
5.147.2.5 speed_1g_fdx . . . . .	302
5.147.2.6 speed_1g_hdx . . . . .	302
5.147.2.7 symmetric_pause . . . . .	302
5.147.2.8 asymmetric_pause . . . . .	303
5.147.2.9 tx_remote_fault . . . . .	303
5.148vtss_phy_clock_conf_t Struct Reference . . . . .	303
5.148.1 Detailed Description . . . . .	303
5.148.2 Field Documentation . . . . .	303
5.148.2.1 src . . . . .	304
5.148.2.2 freq . . . . .	304
5.148.2.3 squelch . . . . .	304
5.149vtss_phy_conf_1g_t Struct Reference . . . . .	304
5.149.1 Detailed Description . . . . .	305
5.149.2 Field Documentation . . . . .	305
5.149.2.1 cfg . . . . .	305
5.149.2.2 val . . . . .	305

5.149.2.3 master . . . . .	305
5.150vtss_phy_conf_t Struct Reference . . . . .	305
5.150.1 Detailed Description . . . . .	306
5.150.2 Field Documentation . . . . .	306
5.150.2.1 mode . . . . .	306
5.150.2.2 forced . . . . .	306
5.150.2.3 aneg . . . . .	306
5.150.2.4 mdi . . . . .	307
5.150.2.5 flf . . . . .	307
5.150.2.6 sigdet . . . . .	307
5.150.2.7 unidir . . . . .	307
5.150.2.8 mac_if_pcs . . . . .	307
5.150.2.9 media_if_pcs . . . . .	308
5.150.2.10force_ams_sel . . . . .	308
5.151vtss_phy_daisy_chain_conf_t Struct Reference . . . . .	308
5.151.1 Detailed Description . . . . .	308
5.151.2 Field Documentation . . . . .	308
5.151.2.1 spi_daisy_input . . . . .	309
5.151.2.2 spi_daisy_output . . . . .	309
5.152vtss_phy_eee_conf_t Struct Reference . . . . .	309
5.152.1 Detailed Description . . . . .	309
5.152.2 Field Documentation . . . . .	309
5.152.2.1 eee_mode . . . . .	310
5.152.2.2 eee_ena_phy . . . . .	310
5.153vtss_phy_enhanced_led_control_t Struct Reference . . . . .	310
5.153.1 Detailed Description . . . . .	310
5.153.2 Field Documentation . . . . .	310
5.153.2.1 ser_led_output_1 . . . . .	311
5.153.2.2 ser_led_output_2 . . . . .	311
5.153.2.3 ser_led_frame_rate . . . . .	311

5.153.2.4 <code>ser_led_select</code>	311
5.154 <code>vtss_phy_forced_t</code> Struct Reference	311
5.154.1 Detailed Description	312
5.154.2 Field Documentation	312
5.154.2.1 <code>speed</code>	312
5.154.2.2 <code>fdx</code>	312
5.155 <code>vtss_phy_led_mode_select_t</code> Struct Reference	312
5.155.1 Detailed Description	313
5.155.2 Field Documentation	313
5.155.2.1 <code>mode</code>	313
5.155.2.2 <code>number</code>	313
5.156 <code>vtss_phy_loopback_t</code> Struct Reference	313
5.156.1 Detailed Description	314
5.156.2 Field Documentation	314
5.156.2.1 <code>far_end_enable</code>	314
5.156.2.2 <code>near_end_enable</code>	314
5.156.2.3 <code>connector_enable</code>	314
5.156.2.4 <code>mac_serdes_input_enable</code>	314
5.156.2.5 <code>mac_serdes_facility_enable</code>	315
5.156.2.6 <code>mac_serdes_equipment_enable</code>	315
5.156.2.7 <code>media_serdes_input_enable</code>	315
5.156.2.8 <code>media_serdes_facility_enable</code>	315
5.156.2.9 <code>media_serdes_equipment_enable</code>	315
5.157 <code>vtss_phy_ltc_freq_synth_s</code> Struct Reference	316
5.157.1 Detailed Description	316
5.157.2 Field Documentation	316
5.157.2.1 <code>enable</code>	316
5.157.2.2 <code>high_duty_cycle</code>	316
5.157.2.3 <code>low_duty_cycle</code>	317
5.158 <code>vtss_phy_mac_serd_pcs_ctrl_t</code> Struct Reference	317

5.158.1 Detailed Description . . . . .	317
5.158.2 Field Documentation . . . . .	317
5.158.2.1 disable . . . . .	318
5.158.2.2 restart . . . . .	318
5.158.2.3 pd_enable . . . . .	318
5.158.2.4 aneg_restart . . . . .	318
5.158.2.5 force_adv_ability . . . . .	318
5.158.2.6 sgmii_in_pre . . . . .	319
5.158.2.7 sgmii_out_pre . . . . .	319
5.158.2.8 serdes_aneg_ena . . . . .	319
5.158.2.9 serdes_pol_inv_in . . . . .	319
5.158.2.10 serdes_pol_inv_out . . . . .	319
5.158.2.11 fast_link_stat_ena . . . . .	320
5.158.2.12 inhibit_odd_start . . . . .	320
5.159vtss_phy_media_serdes_cntl_t Struct Reference . . . . .	320
5.159.1 Detailed Description . . . . .	320
5.159.2 Field Documentation . . . . .	321
5.159.2.1 remote_fault . . . . .	321
5.159.2.2 aneg_pd_detect . . . . .	321
5.159.2.3 force_adv_ability . . . . .	321
5.159.2.4 serdes_pol_inv_in . . . . .	321
5.159.2.5 serdes_pol_inv_out . . . . .	321
5.159.2.6 inhibit_odd_start . . . . .	322
5.159.2.7 force_hls . . . . .	322
5.159.2.8 force_fefi . . . . .	322
5.159.2.9 force_fefi_value . . . . .	322
5.160vtss_phy_pcs_cnt_t Struct Reference . . . . .	322
5.160.1 Detailed Description . . . . .	323
5.160.2 Field Documentation . . . . .	323
5.160.2.1 block_lock_latched . . . . .	323

5.160.2.2 <code>high_ber_latched</code>	323
5.160.2.3 <code>ber_cnt</code>	323
5.160.2.4 <code>err_blk_cnt</code>	324
5.161 <code>vtss_phy_power_conf_t</code> Struct Reference	324
5.161.1 Detailed Description	324
5.161.2 Field Documentation	324
5.161.2.1 <code>mode</code>	324
5.162 <code>vtss_phy_power_status_t</code> Struct Reference	325
5.162.1 Detailed Description	325
5.162.2 Field Documentation	325
5.162.2.1 <code>level</code>	325
5.163 <code>vtss_phy_reset_conf_t</code> Struct Reference	325
5.163.1 Detailed Description	326
5.163.2 Field Documentation	326
5.163.2.1 <code>mac_if</code>	326
5.163.2.2 <code>media_if</code>	326
5.163.2.3 <code>rgmii</code>	326
5.163.2.4 <code>tbi</code>	326
5.163.2.5 <code>force</code>	327
5.163.2.6 <code>pkt_mode</code>	327
5.163.2.7 <code>i_cpu_en</code>	327
5.164 <code>vtss_phy_rgmii_conf_t</code> Struct Reference	327
5.164.1 Detailed Description	327
5.164.2 Field Documentation	328
5.164.2.1 <code>rx_clk_skew_ps</code>	328
5.164.2.2 <code>tx_clk_skew_ps</code>	328
5.165 <code>vtss_phy_statistic_t</code> Struct Reference	328
5.165.1 Detailed Description	328
5.165.2 Field Documentation	329
5.165.2.1 <code>cu_good</code>	329

5.165.2.2 cu_bad . . . . .	329
5.165.2.3 serdes_tx_good . . . . .	329
5.165.2.4 serdes_tx_bad . . . . .	329
5.165.2.5 rx_err_cnt_base_tx . . . . .	330
5.165.2.6 media_mac_serdes_good . . . . .	330
5.165.2.7 media_mac_serdes_crc . . . . .	330
5.166vtss_phy_status_1g_t Struct Reference . . . . .	330
5.166.1 Detailed Description . . . . .	331
5.166.2 Field Documentation . . . . .	331
5.166.2.1 master_cfg_fault . . . . .	331
5.166.2.2 master . . . . .	331
5.167vtss_phy_tbi_conf_t Struct Reference . . . . .	331
5.167.1 Detailed Description . . . . .	331
5.167.2 Field Documentation . . . . .	332
5.167.2.1 aneg_enable . . . . .	332
5.168vtss_phy_timestamp_t Struct Reference . . . . .	332
5.168.1 Detailed Description . . . . .	332
5.168.2 Field Documentation . . . . .	332
5.168.2.1 high . . . . .	332
5.168.2.2 low . . . . .	333
5.168.2.3 seconds . . . . .	333
5.168.2.4 nanoseconds . . . . .	333
5.169vtss_phy_ts_ach_conf_t Struct Reference . . . . .	333
5.169.1 Detailed Description . . . . .	334
5.169.2 Field Documentation . . . . .	334
5.169.2.1 value [1/2] . . . . .	334
5.169.2.2 mask [1/2] . . . . .	334
5.169.2.3 version . . . . .	334
5.169.2.4 value [2/2] . . . . .	334
5.169.2.5 mask [2/2] . . . . .	335

5.169.2.6 channel_type . . . . .	335
5.169.2.7 proto_id . . . . .	335
5.169.2.8 comm_opt . . . . .	335
5.170vtss_phy_ts_alt_clock_mode_s Struct Reference . . . . .	335
5.170.1 Detailed Description . . . . .	336
5.170.2 Field Documentation . . . . .	336
5.170.2.1 pps_ls_lpbk . . . . .	336
5.170.2.2 ls_lpbk . . . . .	336
5.170.2.3 ls_pps_lpbk . . . . .	336
5.171vtss_phy_ts_eng_init_conf_t Struct Reference . . . . .	336
5.171.1 Detailed Description . . . . .	337
5.171.2 Field Documentation . . . . .	337
5.171.2.1 eng_used . . . . .	337
5.171.2.2 encaps_type . . . . .	337
5.171.2.3 flow_match_mode . . . . .	337
5.171.2.4 flow_st_index . . . . .	338
5.171.2.5 flow_end_index . . . . .	338
5.172vtss_phy_ts_engine_action_t Struct Reference . . . . .	338
5.172.1 Detailed Description . . . . .	338
5.172.2 Field Documentation . . . . .	339
5.172.2.1 action_ptp . . . . .	339
5.172.2.2 action_gen . . . . .	339
5.172.2.3 ptp_conf . . . . .	339
5.172.2.4 oam_conf . . . . .	339
5.172.2.5 gen_conf . . . . .	339
5.172.2.6 action . . . . .	340
5.173vtss_phy_ts_engine_flow_conf_t Struct Reference . . . . .	340
5.173.1 Detailed Description . . . . .	340
5.173.2 Field Documentation . . . . .	340
5.173.2.1 eng_mode . . . . .	340

---

5.173.2.2 channel_map . . . . .	341
5.173.2.3 ptp . . . . .	341
5.173.2.4 oam . . . . .	341
5.173.2.5 gen . . . . .	341
5.173.2.6 flow_conf . . . . .	341
5.174vtss_phy_ts_eth_conf_t Struct Reference . . . . .	342
5.174.1 Detailed Description . . . . .	342
5.174.2 Field Documentation . . . . .	343
5.174.2.1 pbb_en . . . . .	343
5.174.2.2 etype . . . . .	343
5.174.2.3 tpid . . . . .	343
5.174.2.4 comm_opt . . . . .	343
5.174.2.5 flow_en . . . . .	344
5.174.2.6 addr_match_mode . . . . .	344
5.174.2.7 addr_match_select . . . . .	344
5.174.2.8 mac_addr . . . . .	344
5.174.2.9 vlan_check . . . . .	344
5.174.2.10num_tag . . . . .	345
5.174.2.11outer_tag_type . . . . .	345
5.174.2.12inner_tag_type . . . . .	345
5.174.2.13tag_range_mode . . . . .	345
5.174.2.14upper . . . . .	345
5.174.2.15lower . . . . .	346
5.174.2.16range [1/2] . . . . .	346
5.174.2.17val [1/2] . . . . .	346
5.174.2.18mask [1/2] . . . . .	346
5.174.2.19value [1/2] . . . . .	346
5.174.2.20outer_tag . . . . .	346
5.174.2.21range [2/2] . . . . .	347
5.174.2.22value [2/2] . . . . .	347

5.174.2.23	val [2/2] . . . . .	347
5.174.2.24	mask [2/2] . . . . .	347
5.174.2.25	_tag . . . . .	347
5.174.2.26	nner_tag . . . . .	347
5.174.2.27	flow_opt . . . . .	348
5.175	vtss_phy_ts_fifo_conf_t Struct Reference . . . . .	348
5.175.1	Detailed Description . . . . .	348
5.175.2	Field Documentation . . . . .	348
5.175.2.1	detect_only . . . . .	348
5.175.2.2	eng_recov . . . . .	349
5.175.2.3	eng_minE . . . . .	349
5.175.2.4	skip_rev_check . . . . .	349
5.176	vtss_phy_ts_fifo_sig_t Struct Reference . . . . .	349
5.176.1	Detailed Description . . . . .	350
5.176.2	Field Documentation . . . . .	350
5.176.2.1	sig_mask . . . . .	350
5.176.2.2	msg_type . . . . .	350
5.176.2.3	domain_num . . . . .	350
5.176.2.4	src_port_identity . . . . .	350
5.176.2.5	sequence_id . . . . .	351
5.176.2.6	dest_ip . . . . .	351
5.176.2.7	src_ip . . . . .	351
5.176.2.8	dest_mac . . . . .	351
5.177	vtss_phy_ts_gen_conf_t Struct Reference . . . . .	351
5.177.1	Detailed Description . . . . .	352
5.177.2	Field Documentation . . . . .	352
5.177.2.1	flow_offset . . . . .	352
5.177.2.2	next_prot_offset . . . . .	352
5.177.2.3	comm_opt . . . . .	352
5.177.2.4	flow_en . . . . .	353

5.177.2.5 data . . . . .	353
5.177.2.6 mask . . . . .	353
5.177.2.7 flow_opt . . . . .	353
5.178vtss_phy_ts_generic_action_t Struct Reference . . . . .	353
5.178.1 Detailed Description . . . . .	354
5.178.2 Field Documentation . . . . .	354
5.178.2.1 enable . . . . .	354
5.178.2.2 channel_map . . . . .	354
5.178.2.3 flow_id . . . . .	354
5.178.2.4 data . . . . .	355
5.178.2.5 mask . . . . .	355
5.178.2.6 ts_type . . . . .	355
5.178.2.7 ts_offset . . . . .	355
5.179vtss_phy_ts_generic_flow_conf_t Struct Reference . . . . .	355
5.179.1 Detailed Description . . . . .	356
5.179.2 Field Documentation . . . . .	356
5.179.2.1 eth1_opt . . . . .	356
5.179.2.2 gen_opt . . . . .	356
5.180vtss_phy_ts_ietf_mpls_ach_oam_conf_t Struct Reference . . . . .	356
5.180.1 Detailed Description . . . . .	357
5.180.2 Field Documentation . . . . .	357
5.180.2.1 delaym_type . . . . .	357
5.180.2.2 ts_format . . . . .	357
5.180.2.3 ds . . . . .	357
5.181vtss_phy_ts_init_conf_t Struct Reference . . . . .	357
5.181.1 Detailed Description . . . . .	358
5.181.2 Field Documentation . . . . .	358
5.181.2.1 clk_freq . . . . .	358
5.181.2.2 clk_src . . . . .	358
5.181.2.3 rx_ts_pos . . . . .	359

---

5.181.2.4 rx_ts_len . . . . .	359
5.181.2.5 tx_fifo_mode . . . . .	359
5.181.2.6 tx_ts_len . . . . .	359
5.181.2.7 tx_fifo_spi_conf . . . . .	359
5.181.2.8 tx_fifo_hi_clk_cycs . . . . .	360
5.181.2.9 tx_fifo_lo_clk_cycs . . . . .	360
5.181.2.10xaui_sel_8487 . . . . .	360
5.181.2.11tc_op_mode . . . . .	360
5.181.2.12auto_clear_ls . . . . .	360
5.181.2.13macsec_ena . . . . .	361
5.181.2.14chk_ing_modified . . . . .	361
5.181.2.15one_step_txfifo . . . . .	361
5.182vtss_phy_ts_ip_conf_t Struct Reference . . . . .	361
5.182.1 Detailed Description . . . . .	362
5.182.2 Field Documentation . . . . .	362
5.182.2.1 ip_mode . . . . .	362
5.182.2.2 sport_val . . . . .	363
5.182.2.3 sport_mask . . . . .	363
5.182.2.4 dport_val . . . . .	363
5.182.2.5 dport_mask . . . . .	363
5.182.2.6 comm_opt . . . . .	363
5.182.2.7 flow_en . . . . .	364
5.182.2.8 match_mode . . . . .	364
5.182.2.9 addr . . . . .	364
5.182.2.10mask . . . . .	364
5.182.2.11ipv4 . . . . .	364
5.182.2.12pv6 . . . . .	365
5.182.2.13p_addr . . . . .	365
5.182.2.14flow_opt . . . . .	365
5.183vtss_phy_ts_mpls_conf_t Struct Reference . . . . .	365

---

5.183.1 Detailed Description . . . . .	366
5.183.2 Field Documentation . . . . .	366
5.183.2.1 cw_en . . . . .	366
5.183.2.2 comm_opt . . . . .	366
5.183.2.3 flow_en . . . . .	366
5.183.2.4 stack_depth . . . . .	366
5.183.2.5 stack_ref_point . . . . .	367
5.183.2.6 top . . . . .	367
5.183.2.7 frst_lvl_after_top . . . . .	367
5.183.2.8 snd_lvl_after_top . . . . .	367
5.183.2.9 thrd_lvl_after_top . . . . .	367
5.183.2.10 op_down . . . . .	368
5.183.2.11 end . . . . .	368
5.183.2.12 frst_lvl_before_end . . . . .	368
5.183.2.13 snd_lvl_before_end . . . . .	368
5.183.2.14 hrd_lvl_before_end . . . . .	368
5.183.2.15 bottom_up . . . . .	368
5.183.2.16 stack_level . . . . .	369
5.183.2.17 flow_opt . . . . .	369
5.184vtss_phy_ts_mpls_lvl_rng_t Struct Reference . . . . .	369
5.184.1 Detailed Description . . . . .	369
5.184.2 Field Documentation . . . . .	369
5.184.2.1 lower . . . . .	369
5.184.2.2 upper . . . . .	370
5.185vtss_phy_ts_nphase_status_t Struct Reference . . . . .	370
5.185.1 Detailed Description . . . . .	370
5.185.2 Field Documentation . . . . .	370
5.185.2.1 enable . . . . .	370
5.185.2.2 CALIB_ERR . . . . .	371
5.185.2.3 CALIB_DONE . . . . .	371

---

5.186vtss_phy_ts_oam_engine_action_t Struct Reference . . . . .	371
5.186.1 Detailed Description . . . . .	371
5.186.2 Field Documentation . . . . .	372
5.186.2.1 enable . . . . .	372
5.186.2.2 y1731_en . . . . .	372
5.186.2.3 channel_map . . . . .	372
5.186.2.4 version . . . . .	372
5.186.2.5 y1731_oam_conf . . . . .	372
5.186.2.6 ietf_oam_conf . . . . .	373
5.186.2.7 oam_conf . . . . .	373
5.187vtss_phy_ts_oam_engine_flow_conf_t Struct Reference . . . . .	373
5.187.1 Detailed Description . . . . .	373
5.187.2 Field Documentation . . . . .	373
5.187.2.1 eth1_opt . . . . .	374
5.187.2.2 eth2_opt . . . . .	374
5.187.2.3 mpls_opt . . . . .	374
5.187.2.4 ach_opt . . . . .	374
5.188vtss_phy_ts_pps_config_s Struct Reference . . . . .	374
5.188.1 Detailed Description . . . . .	375
5.188.2 Field Documentation . . . . .	375
5.188.2.1 pps_width_adj . . . . .	375
5.188.2.2 pps_offset . . . . .	375
5.188.2.3 pps_output_enable . . . . .	375
5.189vtss_phy_ts_ptp_conf_t Struct Reference . . . . .	376
5.189.1 Detailed Description . . . . .	376
5.189.2 Field Documentation . . . . .	376
5.189.2.1 range_en . . . . .	376
5.189.2.2 val . . . . .	376
5.189.2.3 mask . . . . .	377
5.189.2.4 value . . . . .	377

5.189.2.5 upper . . . . .	377
5.189.2.6 lower . . . . .	377
5.189.2.7 range . . . . .	377
5.189.2.8 domain . . . . .	377
5.190vtss_phy_ts_ptp_engine_action_t Struct Reference . . . . .	378
5.190.1 Detailed Description . . . . .	378
5.190.2 Field Documentation . . . . .	378
5.190.2.1 enable . . . . .	378
5.190.2.2 channel_map . . . . .	378
5.190.2.3 ptp_conf . . . . .	379
5.190.2.4 clk_mode . . . . .	379
5.190.2.5 delaym_type . . . . .	379
5.190.2.6 cf_update . . . . .	379
5.191vtss_phy_ts_ptp_engine_flow_conf_t Struct Reference . . . . .	379
5.191.1 Detailed Description . . . . .	380
5.191.2 Field Documentation . . . . .	380
5.191.2.1 eth1_opt . . . . .	380
5.191.2.2 eth2_opt . . . . .	380
5.191.2.3 ip1_opt . . . . .	380
5.191.2.4 ip2_opt . . . . .	381
5.191.2.5 mpls_opt . . . . .	381
5.191.2.6 ach_opt . . . . .	381
5.192vtss_phy_ts_sertod_conf_t Struct Reference . . . . .	381
5.192.1 Detailed Description . . . . .	381
5.192.2 Field Documentation . . . . .	382
5.192.2.1 ip_enable . . . . .	382
5.192.2.2 op_enable . . . . .	382
5.192.2.3 ls_inv . . . . .	382
5.193vtss_phy_ts_stats_t Struct Reference . . . . .	382
5.193.1 Detailed Description . . . . .	383

---

5.193.2 Field Documentation . . . . .	383
5.193.2.1 ingr_pream_shrink_err . . . . .	383
5.193.2.2 egr_pream_shrink_err . . . . .	383
5.193.2.3 ingr_fcs_err . . . . .	383
5.193.2.4 egr_fcs_err . . . . .	384
5.193.2.5 ingr_frm_mod_cnt . . . . .	384
5.193.2.6 egr_frm_mod_cnt . . . . .	384
5.193.2.7 ts_fifo_tx_cnt . . . . .	384
5.193.2.8 ts_fifo_drop_cnt . . . . .	384
5.194vtss_phy_ts_y1731_oam_conf_t Struct Reference . . . . .	385
5.194.1 Detailed Description . . . . .	385
5.194.2 Field Documentation . . . . .	385
5.194.2.1 range_en . . . . .	385
5.194.2.2 delaym_type . . . . .	385
5.194.2.3 val . . . . .	386
5.194.2.4 mask . . . . .	386
5.194.2.5 value . . . . .	386
5.194.2.6 upper . . . . .	386
5.194.2.7 lower . . . . .	386
5.194.2.8 range . . . . .	386
5.194.2.9 meg_level . . . . .	387
5.195vtss_phy_type_t Struct Reference . . . . .	387
5.195.1 Detailed Description . . . . .	387
5.195.2 Field Documentation . . . . .	387
5.195.2.1 part_number . . . . .	387
5.195.2.2 revision . . . . .	388
5.195.2.3 port_cnt . . . . .	388
5.195.2.4 channel_id . . . . .	388
5.195.2.5 base_port_no . . . . .	388
5.195.2.6 phy_api_base_no . . . . .	388

5.196vtss_phy_veriphy_result_t Struct Reference . . . . .	389
5.196.1 Detailed Description . . . . .	389
5.196.2 Field Documentation . . . . .	389
5.196.2.1 link . . . . .	389
5.196.2.2 status . . . . .	389
5.196.2.3 length . . . . .	390
5.197vtss_phy_wol_conf_t Struct Reference . . . . .	390
5.197.1 Detailed Description . . . . .	390
5.197.2 Field Documentation . . . . .	390
5.197.2.1 secure_on_enable . . . . .	390
5.197.2.2 wol_mac . . . . .	391
5.197.2.3 wol_pass . . . . .	391
5.197.2.4 wol_passwd_len . . . . .	391
5.197.2.5 magic_pkt_cnt . . . . .	391
5.198vtss_pi_conf_t Struct Reference . . . . .	391
5.198.1 Detailed Description . . . . .	392
5.198.2 Field Documentation . . . . .	392
5.198.2.1 cs_wait_ns . . . . .	392
5.199vtss_policer_ext_t Struct Reference . . . . .	392
5.199.1 Detailed Description . . . . .	393
5.199.2 Field Documentation . . . . .	393
5.199.2.1 frame_rate . . . . .	393
5.199.2.2 dp_bypass_level . . . . .	393
5.199.2.3 known_unicast . . . . .	393
5.199.2.4 known_multicast . . . . .	393
5.199.2.5 known_broadcast . . . . .	394
5.199.2.6 unknown_unicast . . . . .	394
5.199.2.7 unknown_multicast . . . . .	394
5.199.2.8 unknown_broadcast . . . . .	394
5.199.2.9 learning . . . . .	394

5.199.2.10	to_cpu	395
5.199.2.11	cpu_queue	395
5.199.2.12	limit_noncpu_traffic	395
5.199.2.13	limit_cpu_traffic	395
5.199.2.14	flow_control	395
5.200	vtss_policer_t Struct Reference	396
5.200.1	Detailed Description	396
5.200.2	Field Documentation	396
5.200.2.1	level	396
5.200.2.2	rate	396
5.201	vtss_port_bridge_counters_t Struct Reference	396
5.201.1	Detailed Description	397
5.201.2	Field Documentation	397
5.201.2.1	dot1dTpPortInDiscards	397
5.202	vtss_port_clause_37_adv_t Struct Reference	397
5.202.1	Detailed Description	397
5.202.2	Field Documentation	398
5.202.2.1	fdx	398
5.202.2.2	hdx	398
5.202.2.3	symmetric_pause	398
5.202.2.4	asymmetric_pause	398
5.202.2.5	remote_fault	398
5.202.2.6	acknowledge	399
5.202.2.7	next_page	399
5.203	vtss_port_clause_37_control_t Struct Reference	399
5.203.1	Detailed Description	399
5.203.2	Field Documentation	399
5.203.2.1	enable	400
5.203.2.2	advertisement	400
5.204	vtss_port_conf_t Struct Reference	400

5.204.1 Detailed Description . . . . .	401
5.204.2 Field Documentation . . . . .	401
5.204.2.1 if_type . . . . .	401
5.204.2.2 sd_enable . . . . .	401
5.204.2.3 sd_active_high . . . . .	401
5.204.2.4 sd_internal . . . . .	401
5.204.2.5 frame_gaps . . . . .	402
5.204.2.6 power_down . . . . .	402
5.204.2.7 speed . . . . .	402
5.204.2.8 fdx . . . . .	402
5.204.2.9 flow_control . . . . .	402
5.204.2.10max_frame_length . . . . .	403
5.204.2.11frame_length_chk . . . . .	403
5.204.2.12max_tags . . . . .	403
5.204.2.13exc_col_cont . . . . .	403
5.204.2.14xaui_rx_lane_flip . . . . .	403
5.204.2.15xaui_tx_lane_flip . . . . .	404
5.204.2.16oop . . . . .	404
5.204.2.17serdes . . . . .	404
5.205vtss_port_counters_t Struct Reference . . . . .	404
5.205.1 Detailed Description . . . . .	405
5.205.2 Field Documentation . . . . .	405
5.205.2.1 rmon . . . . .	405
5.205.2.2 if_group . . . . .	405
5.205.2.3 ethernet_like . . . . .	405
5.205.2.4 bridge . . . . .	405
5.205.2.5 prop . . . . .	406
5.206vtss_port_ethernet_like_counters_t Struct Reference . . . . .	406
5.206.1 Detailed Description . . . . .	406
5.206.2 Field Documentation . . . . .	406

---

5.206.2.1 dot3StatsAlignmentErrors . . . . .	407
5.206.2.2 dot3StatsFCSErrors . . . . .	407
5.206.2.3 dot3StatsFrameTooLongs . . . . .	407
5.206.2.4 dot3StatsSymbolErrors . . . . .	407
5.206.2.5 dot3ControlInUnknownOpcodes . . . . .	407
5.206.2.6 dot3InPauseFrames . . . . .	408
5.206.2.7 dot3StatsSingleCollisionFrames . . . . .	408
5.206.2.8 dot3StatsMultipleCollisionFrames . . . . .	408
5.206.2.9 dot3StatsDeferredTransmissions . . . . .	408
5.206.2.10 dot3StatsLateCollisions . . . . .	408
5.206.2.11 dot3StatsExcessiveCollisions . . . . .	409
5.206.2.12 dot3StatsCarrierSenseErrors . . . . .	409
5.206.2.13 dot3OutPauseFrames . . . . .	409
5.207vtss_port_flow_control_conf_t Struct Reference . . . . .	409
5.207.1 Detailed Description . . . . .	410
5.207.2 Field Documentation . . . . .	410
5.207.2.1 obey . . . . .	410
5.207.2.2 generate . . . . .	410
5.207.2.3 smac . . . . .	410
5.207.2.4 pfc . . . . .	410
5.208vtss_port_frame_gaps_t Struct Reference . . . . .	411
5.208.1 Detailed Description . . . . .	411
5.208.2 Field Documentation . . . . .	411
5.208.2.1 hdx_gap_1 . . . . .	411
5.208.2.2 hdx_gap_2 . . . . .	411
5.208.2.3 fdx_gap . . . . .	412
5.209vtss_port_if_group_counters_t Struct Reference . . . . .	412
5.209.1 Detailed Description . . . . .	412
5.209.2 Field Documentation . . . . .	412
5.209.2.1 ifInOctets . . . . .	413

5.209.2.2 ifInUcastPkts . . . . .	413
5.209.2.3 ifInMulticastPkts . . . . .	413
5.209.2.4 ifInBroadcastPkts . . . . .	413
5.209.2.5 ifInNUcastPkts . . . . .	413
5.209.2.6 ifInDiscards . . . . .	414
5.209.2.7 ifInErrors . . . . .	414
5.209.2.8 ifOutOctets . . . . .	414
5.209.2.9 ifOutUcastPkts . . . . .	414
5.209.2.10fOutMulticastPkts . . . . .	414
5.209.2.11fOutBroadcastPkts . . . . .	415
5.209.2.12fOutNUcastPkts . . . . .	415
5.209.2.13fOutDiscards . . . . .	415
5.209.2.14fOutErrors . . . . .	415
5.210vtss_port_ifh_t Struct Reference . . . . .	415
5.210.1 Detailed Description . . . . .	416
5.210.2 Field Documentation . . . . .	416
5.210.2.1 ena_inj_header . . . . .	416
5.210.2.2 ena_xtr_header . . . . .	416
5.210.2.3 ena_ifh_header . . . . .	416
5.211vtss_port_map_t Struct Reference . . . . .	417
5.211.1 Detailed Description . . . . .	417
5.211.2 Field Documentation . . . . .	417
5.211.2.1 chip_port . . . . .	417
5.211.2.2 chip_no . . . . .	417
5.211.2.3 max_bw . . . . .	418
5.211.2.4 miim_controller . . . . .	418
5.211.2.5 miim_addr . . . . .	418
5.211.2.6 miim_chip_no . . . . .	418
5.212vtss_port_proprietary_counters_t Struct Reference . . . . .	418
5.212.1 Detailed Description . . . . .	419

5.212.2 Field Documentation . . . . .	419
5.212.2.1 rx_prio . . . . .	419
5.212.2.2 tx_prio . . . . .	419
5.213vtss_port_rmon_counters_t Struct Reference . . . . .	419
5.213.1 Detailed Description . . . . .	420
5.213.2 Field Documentation . . . . .	420
5.213.2.1 rx_etherStatsDropEvents . . . . .	420
5.213.2.2 rx_etherStatsOctets . . . . .	421
5.213.2.3 rx_etherStatsPkts . . . . .	421
5.213.2.4 rx_etherStatsBroadcastPkts . . . . .	421
5.213.2.5 rx_etherStatsMulticastPkts . . . . .	421
5.213.2.6 rx_etherStatsCRCAlignErrors . . . . .	421
5.213.2.7 rx_etherStatsUndersizePkts . . . . .	422
5.213.2.8 rx_etherStatsOversizePkts . . . . .	422
5.213.2.9 rx_etherStatsFragments . . . . .	422
5.213.2.10x_etherStatsJabbers . . . . .	422
5.213.2.11rx_etherStatsPkts64Octets . . . . .	422
5.213.2.12rx_etherStatsPkts65to127Octets . . . . .	423
5.213.2.13rx_etherStatsPkts128to255Octets . . . . .	423
5.213.2.14rx_etherStatsPkts256to511Octets . . . . .	423
5.213.2.15rx_etherStatsPkts512to1023Octets . . . . .	423
5.213.2.16rx_etherStatsPkts1024to1518Octets . . . . .	423
5.213.2.17rx_etherStatsPkts1519toMaxOctets . . . . .	424
5.213.2.18x_etherStatsDropEvents . . . . .	424
5.213.2.19x_etherStatsOctets . . . . .	424
5.213.2.20tx_etherStatsPkts . . . . .	424
5.213.2.21tx_etherStatsBroadcastPkts . . . . .	424
5.213.2.22tx_etherStatsMulticastPkts . . . . .	425
5.213.2.23tx_etherStatsCollisions . . . . .	425
5.213.2.24tx_etherStatsPkts64Octets . . . . .	425

5.213.2.25x_etherStatsPkts65to127Octets . . . . .	425
5.213.2.26x_etherStatsPkts128to255Octets . . . . .	425
5.213.2.27x_etherStatsPkts256to511Octets . . . . .	426
5.213.2.28x_etherStatsPkts512to1023Octets . . . . .	426
5.213.2.29x_etherStatsPkts1024to1518Octets . . . . .	426
5.213.2.30x_etherStatsPkts1519toMaxOctets . . . . .	426
5.214vtss_port_sedes_conf_t Struct Reference . . . . .	426
5.214.1 Detailed Description . . . . .	427
5.214.2 Field Documentation . . . . .	427
5.214.2.1 sfp_dac . . . . .	427
5.215vtss_port_sgmii_aneg_t Struct Reference . . . . .	427
5.215.1 Detailed Description . . . . .	427
5.215.2 Field Documentation . . . . .	428
5.215.2.1 link . . . . .	428
5.215.2.2 fdx . . . . .	428
5.215.2.3 hdx . . . . .	428
5.215.2.4 speed_10M . . . . .	428
5.215.2.5 speed_100M . . . . .	428
5.215.2.6 speed_1G . . . . .	429
5.215.2.7 aneg_complete . . . . .	429
5.216vtss_port_status_t Struct Reference . . . . .	429
5.216.1 Detailed Description . . . . .	429
5.216.2 Field Documentation . . . . .	430
5.216.2.1 link_down . . . . .	430
5.216.2.2 link . . . . .	430
5.216.2.3 speed . . . . .	430
5.216.2.4 fdx . . . . .	430
5.216.2.5 remote_fault . . . . .	430
5.216.2.6 aneg_complete . . . . .	431
5.216.2.7 unidirectional_ability . . . . .	431

5.216.2.8 aneg . . . . .	431
5.216.2.9 mdi_cross . . . . .	431
5.216.2.10fiber . . . . .	431
5.216.2.11copper . . . . .	432
5.217vtss_qce_action_t Struct Reference . . . . .	432
5.217.1 Detailed Description . . . . .	432
5.217.2 Field Documentation . . . . .	432
5.217.2.1 prio_enable . . . . .	432
5.217.2.2 prio . . . . .	433
5.217.2.3 dp_enable . . . . .	433
5.217.2.4 dp . . . . .	433
5.217.2.5 dscp_enable . . . . .	433
5.217.2.6 dscp . . . . .	433
5.217.2.7 pcp_dei_enable . . . . .	434
5.217.2.8 pcp . . . . .	434
5.217.2.9 dei . . . . .	434
5.217.2.10policy_no_enable . . . . .	434
5.217.2.11policy_no . . . . .	434
5.218vtss_qce_frame_etype_t Struct Reference . . . . .	435
5.218.1 Detailed Description . . . . .	435
5.218.2 Field Documentation . . . . .	435
5.218.2.1 etype . . . . .	435
5.218.2.2 data . . . . .	435
5.219vtss_qce_frame_ipv4_t Struct Reference . . . . .	435
5.219.1 Detailed Description . . . . .	436
5.219.2 Field Documentation . . . . .	436
5.219.2.1 fragment . . . . .	436
5.219.2.2 dscp . . . . .	436
5.219.2.3 proto . . . . .	436
5.219.2.4 sip . . . . .	437

5.219.2.5 dip . . . . .	437
5.219.2.6 sport . . . . .	437
5.219.2.7 dport . . . . .	437
5.220vtss_qce_frame_ipv6_t Struct Reference . . . . .	437
5.220.1 Detailed Description . . . . .	438
5.220.2 Field Documentation . . . . .	438
5.220.2.1 dscp . . . . .	438
5.220.2.2 proto . . . . .	438
5.220.2.3 sip . . . . .	438
5.220.2.4 dip . . . . .	439
5.220.2.5 sport . . . . .	439
5.220.2.6 dport . . . . .	439
5.221vtss_qce_frame_llc_t Struct Reference . . . . .	439
5.221.1 Detailed Description . . . . .	439
5.221.2 Field Documentation . . . . .	440
5.221.2.1 data . . . . .	440
5.222vtss_qce_frame_snap_t Struct Reference . . . . .	440
5.222.1 Detailed Description . . . . .	440
5.222.2 Field Documentation . . . . .	440
5.222.2.1 data . . . . .	440
5.223vtss_qce_key_t Struct Reference . . . . .	441
5.223.1 Detailed Description . . . . .	441
5.223.2 Field Documentation . . . . .	441
5.223.2.1 port_list . . . . .	441
5.223.2.2 mac . . . . .	441
5.223.2.3 tag . . . . .	442
5.223.2.4 inner_tag . . . . .	442
5.223.2.5 type . . . . .	442
5.223.2.6 etype . . . . .	442
5.223.2.7 llc . . . . .	442

5.223.2.8 snap . . . . .	443
5.223.2.9 ipv4 . . . . .	443
5.223.2.10pv6 . . . . .	443
5.223.2.11frame . . . . .	443
5.224vtss_qce_mac_t Struct Reference . . . . .	443
5.224.1 Detailed Description . . . . .	444
5.224.2 Field Documentation . . . . .	444
5.224.2.1 dmac_mc . . . . .	444
5.224.2.2 dmac_bc . . . . .	444
5.224.2.3 dmac . . . . .	444
5.224.2.4 smac . . . . .	445
5.225vtss_qce_t Struct Reference . . . . .	445
5.225.1 Detailed Description . . . . .	445
5.225.2 Field Documentation . . . . .	445
5.225.2.1 id . . . . .	445
5.225.2.2 key . . . . .	446
5.225.2.3 action . . . . .	446
5.226vtss_qce_tag_t Struct Reference . . . . .	446
5.226.1 Detailed Description . . . . .	446
5.226.2 Field Documentation . . . . .	446
5.226.2.1 vid . . . . .	447
5.226.2.2 pcp . . . . .	447
5.226.2.3 dei . . . . .	447
5.226.2.4 tagged . . . . .	447
5.226.2.5 s_tag . . . . .	447
5.227vtss_qos_conf_t Struct Reference . . . . .	448
5.227.1 Detailed Description . . . . .	448
5.227.2 Field Documentation . . . . .	448
5.227.2.1 prios . . . . .	448
5.227.2.2 dscp_trust . . . . .	449

5.227.2.3 dscp_qos_class_map . . . . .	449
5.227.2.4 dscp_dp_level_map . . . . .	449
5.227.2.5 dscp_qos_map . . . . .	449
5.227.2.6 dscp_qos_map_dp1 . . . . .	449
5.227.2.7 dscp_qos_map_dp2 . . . . .	450
5.227.2.8 dscp_qos_map_dp3 . . . . .	450
5.227.2.9 dscp_remark . . . . .	450
5.227.2.10 dscp_translate_map . . . . .	450
5.227.2.11 dscp_remap . . . . .	450
5.227.2.12 policer_uc . . . . .	451
5.227.2.13 policer_uc_frame_rate . . . . .	451
5.227.2.14 policer_uc_mode . . . . .	451
5.227.2.15 policer_mc . . . . .	451
5.227.2.16 policer_mc_frame_rate . . . . .	451
5.227.2.17 policer_mc_mode . . . . .	452
5.227.2.18 policer_bc . . . . .	452
5.227.2.19 policer_bc_frame_rate . . . . .	452
5.227.2.20 policer_bc_mode . . . . .	452
5.227.2.21 red_v3 . . . . .	452
5.228 vtss_qos_port_conf_t Struct Reference . . . . .	453
5.228.1 Detailed Description . . . . .	453
5.228.2 Field Documentation . . . . .	453
5.228.2.1 policer_port . . . . .	453
5.228.2.2 policer_ext_port . . . . .	454
5.228.2.3 policer_queue . . . . .	454
5.228.2.4 shaper_port . . . . .	454
5.228.2.5 shaper_queue . . . . .	454
5.228.2.6 default_prio . . . . .	454
5.228.2.7 usr_prio . . . . .	455
5.228.2.8 default_dpl . . . . .	455

---

5.228.2.9 default_dei . . . . .	455
5.228.2.10 tag_class_enable . . . . .	455
5.228.2.11 qos_class_map . . . . .	455
5.228.2.12 dp_level_map . . . . .	456
5.228.2.13 dscp_class_enable . . . . .	456
5.228.2.14 dscp_mode . . . . .	456
5.228.2.15 dscp_emode . . . . .	456
5.228.2.16 dscp_translate . . . . .	456
5.228.2.17 tag_remark_mode . . . . .	457
5.228.2.18 ag_default_pcp . . . . .	457
5.228.2.19 tag_default_dei . . . . .	457
5.228.2.20 tag_pcp_map . . . . .	457
5.228.2.21 tag_dei_map . . . . .	457
5.228.2.22 dwrr_enable . . . . .	458
5.228.2.23 dwrr_cnt . . . . .	458
5.228.2.24 queue_pct . . . . .	458
5.228.2.25 wred_group . . . . .	458
5.229 vtss_rcpll_status_t Struct Reference . . . . .	458
5.229.1 Detailed Description . . . . .	459
5.229.2 Field Documentation . . . . .	459
5.229.2.1 out_of_range . . . . .	459
5.229.2.2 cal_error . . . . .	459
5.229.2.3 cal_not_done . . . . .	459
5.230 vtss_red_v2_t Struct Reference . . . . .	460
5.230.1 Detailed Description . . . . .	460
5.230.2 Field Documentation . . . . .	460
5.230.2.1 enable . . . . .	460
5.230.2.2 min_fl . . . . .	460
5.230.2.3 max . . . . .	461
5.230.2.4 max_unit . . . . .	461

5.231vtss_restart_status_t Struct Reference . . . . .	461
5.231.1 Detailed Description . . . . .	461
5.231.2 Field Documentation . . . . .	461
5.231.2.1 restart . . . . .	462
5.231.2.2 prev_version . . . . .	462
5.231.2.3 cur_version . . . . .	462
5.232vtss_routing_entry_t Struct Reference . . . . .	462
5.232.1 Detailed Description . . . . .	463
5.232.2 Field Documentation . . . . .	463
5.232.2.1 type . . . . .	463
5.232.2.2 ipv4_uc . . . . .	463
5.232.2.3 ipv6_uc . . . . .	463
5.232.2.4 route . . . . .	463
5.232.2.5 vlan . . . . .	464
5.233vtss_secure_on_passwd_t Struct Reference . . . . .	464
5.233.1 Detailed Description . . . . .	464
5.233.2 Field Documentation . . . . .	464
5.233.2.1 passwd . . . . .	464
5.234vtss_sedes_macro_conf_t Struct Reference . . . . .	465
5.234.1 Detailed Description . . . . .	465
5.234.2 Field Documentation . . . . .	465
5.234.2.1 serdes1g_vdd . . . . .	465
5.234.2.2 serdes6g_vdd . . . . .	465
5.234.2.3 ib_cterm_ena . . . . .	466
5.234.2.4 qsgmii . . . . .	466
5.235vtss_sflow_port_conf_t Struct Reference . . . . .	466
5.235.1 Detailed Description . . . . .	466
5.235.2 Field Documentation . . . . .	466
5.235.2.1 type . . . . .	467
5.235.2.2 sampling_rate . . . . .	467

---

5.236vtss_sgpio_conf_t Struct Reference . . . . .	467
5.236.1 Detailed Description . . . . .	467
5.236.2 Field Documentation . . . . .	467
5.236.2.1 bmode . . . . .	468
5.236.2.2 bit_count . . . . .	468
5.236.2.3 port_conf . . . . .	468
5.237vtss_sgpio_port_conf_t Struct Reference . . . . .	468
5.237.1 Detailed Description . . . . .	468
5.237.2 Field Documentation . . . . .	469
5.237.2.1 enabled . . . . .	469
5.237.2.2 mode . . . . .	469
5.237.2.3 int_pol_high . . . . .	469
5.238vtss_sgpio_port_data_t Struct Reference . . . . .	469
5.238.1 Detailed Description . . . . .	470
5.238.2 Field Documentation . . . . .	470
5.238.2.1 value . . . . .	470
5.239vtss_shaper_t Struct Reference . . . . .	470
5.239.1 Detailed Description . . . . .	470
5.239.2 Field Documentation . . . . .	470
5.239.2.1 level . . . . .	471
5.239.2.2 rate . . . . .	471
5.239.2.3 ebs . . . . .	471
5.239.2.4 eir . . . . .	471
5.240vtss_sublayer_status_t Struct Reference . . . . .	471
5.240.1 Detailed Description . . . . .	472
5.240.2 Field Documentation . . . . .	472
5.240.2.1 rx_link . . . . .	472
5.240.2.2 link_down . . . . .	472
5.240.2.3 rx_fault . . . . .	472
5.240.2.4 tx_fault . . . . .	473

5.241vtss_sync_clock_in_t Struct Reference . . . . .	473
5.241.1 Detailed Description . . . . .	473
5.241.2 Field Documentation . . . . .	473
5.241.2.1 port_no . . . . .	473
5.241.2.2 squelsh . . . . .	474
5.241.2.3 enable . . . . .	474
5.242vtss_sync_clock_out_t Struct Reference . . . . .	474
5.242.1 Detailed Description . . . . .	474
5.242.2 Field Documentation . . . . .	474
5.242.2.1 divider . . . . .	475
5.242.2.2 enable . . . . .	475
5.243vtss_tci_t Struct Reference . . . . .	475
5.243.1 Detailed Description . . . . .	475
5.243.2 Field Documentation . . . . .	475
5.243.2.1 vid . . . . .	476
5.243.2.2 cfi . . . . .	476
5.243.2.3 tagprior . . . . .	476
5.244vtss_timeofday_t Struct Reference . . . . .	476
5.244.1 Detailed Description . . . . .	476
5.244.2 Field Documentation . . . . .	477
5.244.2.1 sec [1/2] . . . . .	477
5.244.2.2 sec [2/2] . . . . .	477
5.245vtss_timestamp_t Struct Reference . . . . .	477
5.245.1 Detailed Description . . . . .	477
5.245.2 Field Documentation . . . . .	478
5.245.2.1 sec_msb . . . . .	478
5.245.2.2 seconds . . . . .	478
5.245.2.3 nanoseconds . . . . .	478
5.246vtss_trace_conf_t Struct Reference . . . . .	478
5.246.1 Detailed Description . . . . .	479

---

5.246.2 Field Documentation . . . . .	479
5.246.2.1 level . . . . .	479
5.247vtss_ts_alt_clock_mode_t Struct Reference . . . . .	479
5.247.1 Detailed Description . . . . .	479
5.247.2 Field Documentation . . . . .	479
5.247.2.1 one_pps_out . . . . .	480
5.247.2.2 one_pps_in . . . . .	480
5.247.2.3 save . . . . .	480
5.247.2.4 load . . . . .	480
5.248vtss_ts_ext_clock_mode_t Struct Reference . . . . .	480
5.248.1 Detailed Description . . . . .	481
5.248.2 Field Documentation . . . . .	481
5.248.2.1 one_pps_mode . . . . .	481
5.248.2.2 enable . . . . .	481
5.248.2.3 freq . . . . .	481
5.249vtss_ts_id_t Struct Reference . . . . .	482
5.249.1 Detailed Description . . . . .	482
5.249.2 Field Documentation . . . . .	482
5.249.2.1 ts_id . . . . .	482
5.250vtss_ts_internal_mode_t Struct Reference . . . . .	482
5.250.1 Detailed Description . . . . .	483
5.250.2 Field Documentation . . . . .	483
5.250.2.1 int_fmt . . . . .	483
5.251vtss_ts_operation_mode_t Struct Reference . . . . .	483
5.251.1 Detailed Description . . . . .	483
5.251.2 Field Documentation . . . . .	483
5.251.2.1 mode . . . . .	484
5.252vtss_ts_timestamp_alloc_t Struct Reference . . . . .	484
5.252.1 Detailed Description . . . . .	484
5.252.2 Field Documentation . . . . .	484

5.252.2.1 port_mask . . . . .	484
5.252.2.2 context . . . . .	485
5.252.2.3 cb . . . . .	485
5.253vtss_ts_timestamp_t Struct Reference . . . . .	485
5.253.1 Detailed Description . . . . .	485
5.253.2 Field Documentation . . . . .	485
5.253.2.1 ts . . . . .	486
5.253.2.2 id . . . . .	486
5.253.2.3 context . . . . .	486
5.253.2.4 ts_valid . . . . .	486
5.254vtss_vcap_ip_t Struct Reference . . . . .	486
5.254.1 Detailed Description . . . . .	487
5.254.2 Field Documentation . . . . .	487
5.254.2.1 value . . . . .	487
5.254.2.2 mask . . . . .	487
5.255vtss_vcap_u128_t Struct Reference . . . . .	487
5.255.1 Detailed Description . . . . .	488
5.255.2 Field Documentation . . . . .	488
5.255.2.1 value . . . . .	488
5.255.2.2 mask . . . . .	488
5.256vtss_vcap_u16_t Struct Reference . . . . .	488
5.256.1 Detailed Description . . . . .	489
5.256.2 Field Documentation . . . . .	489
5.256.2.1 value . . . . .	489
5.256.2.2 mask . . . . .	489
5.257vtss_vcap_u24_t Struct Reference . . . . .	489
5.257.1 Detailed Description . . . . .	490
5.257.2 Field Documentation . . . . .	490
5.257.2.1 value . . . . .	490
5.257.2.2 mask . . . . .	490

---

5.258vtss_vcap_u32_t Struct Reference . . . . .	490
5.258.1 Detailed Description . . . . .	491
5.258.2 Field Documentation . . . . .	491
5.258.2.1 value . . . . .	491
5.258.2.2 mask . . . . .	491
5.259vtss_vcap_u40_t Struct Reference . . . . .	491
5.259.1 Detailed Description . . . . .	492
5.259.2 Field Documentation . . . . .	492
5.259.2.1 value . . . . .	492
5.259.2.2 mask . . . . .	492
5.260vtss_vcap_u48_t Struct Reference . . . . .	492
5.260.1 Detailed Description . . . . .	493
5.260.2 Field Documentation . . . . .	493
5.260.2.1 value . . . . .	493
5.260.2.2 mask . . . . .	493
5.261vtss_vcap_u8_t Struct Reference . . . . .	493
5.261.1 Detailed Description . . . . .	494
5.261.2 Field Documentation . . . . .	494
5.261.2.1 value . . . . .	494
5.261.2.2 mask . . . . .	494
5.262vtss_vcap_udp_tcp_t Struct Reference . . . . .	494
5.262.1 Detailed Description . . . . .	495
5.262.2 Field Documentation . . . . .	495
5.262.2.1 in_range . . . . .	495
5.262.2.2 low . . . . .	495
5.262.2.3 high . . . . .	495
5.263vtss_vcap_vid_t Struct Reference . . . . .	495
5.263.1 Detailed Description . . . . .	496
5.263.2 Field Documentation . . . . .	496
5.263.2.1 value . . . . .	496

5.263.2.2 mask . . . . .	496
5.264vtss_vcap_vr_t Struct Reference . . . . .	496
5.264.1 Detailed Description . . . . .	497
5.264.2 Field Documentation . . . . .	497
5.264.2.1 type . . . . .	497
5.264.2.2 value . . . . .	497
5.264.2.3 mask . . . . .	498
5.264.2.4 v . . . . .	498
5.264.2.5 low . . . . .	498
5.264.2.6 high . . . . .	498
5.264.2.7 r . . . . .	498
5.264.2.8 vr . . . . .	498
5.265vtss_vce_action_t Struct Reference . . . . .	499
5.265.1 Detailed Description . . . . .	499
5.265.2 Field Documentation . . . . .	499
5.265.2.1 vid . . . . .	499
5.265.2.2 policy_no . . . . .	499
5.266vtss_vce_frame_etype_t Struct Reference . . . . .	499
5.266.1 Detailed Description . . . . .	500
5.266.2 Field Documentation . . . . .	500
5.266.2.1 etype . . . . .	500
5.266.2.2 data . . . . .	500
5.267vtss_vce_frame_ipv4_t Struct Reference . . . . .	500
5.267.1 Detailed Description . . . . .	501
5.267.2 Field Documentation . . . . .	501
5.267.2.1 fragment . . . . .	501
5.267.2.2 options . . . . .	501
5.267.2.3 dscp . . . . .	501
5.267.2.4 proto . . . . .	502
5.267.2.5 sip . . . . .	502

5.267.2.6 dport . . . . .	502
5.268vtss_vce_frame_ipv6_t Struct Reference . . . . .	502
5.268.1 Detailed Description . . . . .	503
5.268.2 Field Documentation . . . . .	503
5.268.2.1 dscp . . . . .	503
5.268.2.2 proto . . . . .	503
5.268.2.3 sip . . . . .	503
5.268.2.4 dport . . . . .	503
5.269vtss_vce_frame_llc_t Struct Reference . . . . .	504
5.269.1 Detailed Description . . . . .	504
5.269.2 Field Documentation . . . . .	504
5.269.2.1 data . . . . .	504
5.270vtss_vce_frame_snap_t Struct Reference . . . . .	504
5.270.1 Detailed Description . . . . .	505
5.270.2 Field Documentation . . . . .	505
5.270.2.1 data . . . . .	505
5.271vtss_vce_key_t Struct Reference . . . . .	505
5.271.1 Detailed Description . . . . .	505
5.271.2 Field Documentation . . . . .	506
5.271.2.1 port_list . . . . .	506
5.271.2.2 mac . . . . .	506
5.271.2.3 tag . . . . .	506
5.271.2.4 type . . . . .	506
5.271.2.5 etype . . . . .	506
5.271.2.6 llc . . . . .	507
5.271.2.7 snap . . . . .	507
5.271.2.8 ipv4 . . . . .	507
5.271.2.9 ipv6 . . . . .	507
5.271.2.10 frame . . . . .	507
5.272vtss_vce_mac_t Struct Reference . . . . .	508

5.272.1 Detailed Description . . . . .	508
5.272.2 Field Documentation . . . . .	508
5.272.2.1 dmac_mc . . . . .	508
5.272.2.2 dmac_bc . . . . .	508
5.272.2.3 smac . . . . .	509
5.273vtss_vce_t Struct Reference . . . . .	509
5.273.1 Detailed Description . . . . .	509
5.273.2 Field Documentation . . . . .	509
5.273.2.1 id . . . . .	509
5.273.2.2 key . . . . .	510
5.273.2.3 action . . . . .	510
5.274vtss_vce_tag_t Struct Reference . . . . .	510
5.274.1 Detailed Description . . . . .	510
5.274.2 Field Documentation . . . . .	510
5.274.2.1 vid . . . . .	511
5.274.2.2 pcp . . . . .	511
5.274.2.3 dei . . . . .	511
5.274.2.4 tagged . . . . .	511
5.274.2.5 s_tag . . . . .	511
5.275vtss_vcl_port_conf_t Struct Reference . . . . .	512
5.275.1 Detailed Description . . . . .	512
5.275.2 Field Documentation . . . . .	512
5.275.2.1 dmac_dip . . . . .	512
5.276vtss_vid_mac_t Struct Reference . . . . .	512
5.276.1 Detailed Description . . . . .	513
5.276.2 Field Documentation . . . . .	513
5.276.2.1 vid . . . . .	513
5.276.2.2 mac . . . . .	513
5.277vtss_vlan_conf_t Struct Reference . . . . .	513
5.277.1 Detailed Description . . . . .	513

---

5.277.2 Field Documentation . . . . .	514
5.277.2.1 s_etype . . . . .	514
5.278vtss_vlan_port_conf_t Struct Reference . . . . .	514
5.278.1 Detailed Description . . . . .	514
5.278.2 Field Documentation . . . . .	514
5.278.2.1 port_type . . . . .	514
5.278.2.2 pvid . . . . .	515
5.278.2.3 untagged_vid . . . . .	515
5.278.2.4 frame_type . . . . .	515
5.278.2.5 ingress_filter . . . . .	515
5.279vtss_vlan_tag_t Struct Reference . . . . .	515
5.279.1 Detailed Description . . . . .	516
5.279.2 Field Documentation . . . . .	516
5.279.2.1 tpid . . . . .	516
5.279.2.2 pcp . . . . .	516
5.279.2.3 dei . . . . .	516
5.279.2.4 vid . . . . .	517
5.280vtss_vlan_trans_grp2vlan_conf_t Struct Reference . . . . .	517
5.280.1 Detailed Description . . . . .	517
5.280.2 Field Documentation . . . . .	517
5.280.2.1 group_id . . . . .	517
5.280.2.2 vid . . . . .	518
5.280.2.3 trans_vid . . . . .	518
5.281vtss_vlan_trans_port2grp_conf_t Struct Reference . . . . .	518
5.281.1 Detailed Description . . . . .	518
5.281.2 Field Documentation . . . . .	518
5.281.2.1 group_id . . . . .	519
5.281.2.2 ports . . . . .	519
5.282vtss_vlan_vid_conf_t Struct Reference . . . . .	519
5.282.1 Detailed Description . . . . .	519

5.282.2 Field Documentation . . . . .	519
5.282.2.1 learning . . . . .	520
5.282.2.2 mirror . . . . .	520
5.283vtss_vstax_conf_t Struct Reference . . . . .	520
5.283.1 Detailed Description . . . . .	520
5.283.2 Field Documentation . . . . .	520
5.283.2.1 upsid_0 . . . . .	521
5.283.2.2 upsid_1 . . . . .	521
5.283.2.3 port_0 . . . . .	521
5.283.2.4 port_1 . . . . .	521
5.283.2.5 cmef_disable . . . . .	521
5.284vtss_vstax_glag_entry_t Struct Reference . . . . .	522
5.284.1 Detailed Description . . . . .	522
5.284.2 Field Documentation . . . . .	522
5.284.2.1 upsid . . . . .	522
5.284.2.2 upspn . . . . .	522
5.285vtss_vstax_port_conf_t Struct Reference . . . . .	522
5.285.1 Detailed Description . . . . .	523
5.285.2 Field Documentation . . . . .	523
5.285.2.1 ttl . . . . .	523
5.285.2.2 mirror . . . . .	523
5.286vtss_vstax_route_entry_t Struct Reference . . . . .	523
5.286.1 Detailed Description . . . . .	524
5.286.2 Field Documentation . . . . .	524
5.286.2.1 stack_port_a . . . . .	524
5.286.2.2 stack_port_b . . . . .	524
5.287vtss_vstax_route_table_t Struct Reference . . . . .	524
5.287.1 Detailed Description . . . . .	525
5.287.2 Field Documentation . . . . .	525
5.287.2.1 topology_type . . . . .	525

5.287.2.2 table . . . . .	525
5.288vtss_vstax_rx_header_t Struct Reference . . . . .	525
5.288.1 Detailed Description . . . . .	526
5.288.2 Field Documentation . . . . .	526
5.288.2.1 valid . . . . .	526
5.288.2.2 sp . . . . .	526
5.288.2.3 upsid . . . . .	526
5.288.2.4 port_no . . . . .	526
5.288.2.5 glag_no . . . . .	527
5.288.2.6 isdx . . . . .	527
5.289vtss_vstax_tx_header_t Struct Reference . . . . .	527
5.289.1 Detailed Description . . . . .	527
5.289.2 Field Documentation . . . . .	528
5.289.2.1 fwd_mode . . . . .	528
5.289.2.2 ttl . . . . .	528
5.289.2.3 prio . . . . .	528
5.289.2.4 upsid . . . . .	528
5.289.2.5 tci . . . . .	528
5.289.2.6 port_no . . . . .	529
5.289.2.7 chip_port . . . . .	529
5.289.2.8 glag_no . . . . .	529
5.289.2.9 queue_no . . . . .	529
5.289.2.10keep_ttl . . . . .	529
5.289.2.11dp . . . . .	530
5.290vtss_wol_mac_addr_t Struct Reference . . . . .	530
5.290.1 Detailed Description . . . . .	530
5.290.2 Field Documentation . . . . .	530
5.290.2.1 addr . . . . .	530

<b>6 File Documentation</b>	<b>531</b>
6.1 vtss_api/include/vtss/api/I2_types.h File Reference . . . . .	531
6.1.1 Detailed Description . . . . .	531
6.1.2 Enumeration Type Documentation . . . . .	531
6.1.2.1 vtss_sflow_type_t . . . . .	531
6.2 vtss_api/include/vtss/api/options.h File Reference . . . . .	532
6.2.1 Detailed Description . . . . .	534
6.2.2 Macro Definition Documentation . . . . .	534
6.2.2.1 VTSS_ARCH_JAGUAR_2 . . . . .	534
6.2.2.2 VTSS_ARCH_JAGUAR_2_B . . . . .	534
6.2.2.3 VTSS_CHIP_10G_PHY . . . . .	534
6.2.2.4 VTSS_FEATURE_MISC . . . . .	535
6.2.2.5 VTSS_FEATURE_SERIAL_GPIO . . . . .	535
6.2.2.6 VTSS_FEATURE_PORT_CONTROL . . . . .	535
6.2.2.7 VTSS_FEATURE_PORT_IFH . . . . .	535
6.2.2.8 VTSS_FEATURE_CLAUSE_37 . . . . .	535
6.2.2.9 VTSS_FEATURE_EXC_COL_CONT . . . . .	536
6.2.2.10 VTSS_FEATURE_PORT_CNT_ETHER_LIKE . . . . .	536
6.2.2.11 VTSS_FEATURE_PORT_CNT_BRIDGE . . . . .	536
6.2.2.12 VTSS_FEATURE_QOS . . . . .	536
6.2.2.13 VTSS_FEATURE_VSTAX . . . . .	536
6.2.2.14 VTSS_FEATURE_VSTAX_V2 . . . . .	537
6.2.2.15 VTSS_FEATURE_AGGR_GLAG . . . . .	537
6.2.2.16 VTSS_FEATURE_PORT_MUX . . . . .	537
6.2.2.17 VTSS_FEATURE_PFC . . . . .	537
6.2.2.18 VTSS_FEATURE_QCL . . . . .	537
6.2.2.19 VTSS_FEATURE_QCL_V2 . . . . .	538
6.2.2.20 VTSS_FEATURE_QCL_KEY_S_TAG . . . . .	538
6.2.2.21 VTSS_FEATURE_QCL_KEY_INNER_TAG . . . . .	538
6.2.2.22 VTSS_FEATURE_QCL_KEY_DMAC . . . . .	538

6.2.2.23	VTSS_FEATURE_QCL_KEY_DIP . . . . .	538
6.2.2.24	VTSS_FEATURE_QCL_PCP_DEI_ACTION . . . . .	539
6.2.2.25	VTSS_FEATURE_QCL_POLICY_ACTION . . . . .	539
6.2.2.26	VTSS_FEATURE_QOS_POLICER_UC_SWITCH . . . . .	539
6.2.2.27	VTSS_FEATURE_QOS_POLICER_MC_SWITCH . . . . .	539
6.2.2.28	VTSS_FEATURE_QOS_POLICER_BC_SWITCH . . . . .	539
6.2.2.29	VTSS_FEATURE_QOS_PORT_POLICER_EXT . . . . .	540
6.2.2.30	VTSS_FEATURE_QOS_PORT_POLICER_EXT_FPS . . . . .	540
6.2.2.31	VTSS_FEATURE_QOS_PORT_POLICER_EXT_FC . . . . .	540
6.2.2.32	VTSS_FEATURE_QOS_PORT_POLICER_EXT_DPBL . . . . .	540
6.2.2.33	VTSS_FEATURE_QOS_PORT_POLICER_EXT_TTM_V2 . . . . .	540
6.2.2.34	VTSS_FEATURE_QOS_QUEUE_POLICER . . . . .	541
6.2.2.35	VTSS_FEATURE_QOS_QUEUE_TX . . . . .	541
6.2.2.36	VTSS_FEATURE_QOS_SCHEDULER_V2 . . . . .	541
6.2.2.37	VTSS_FEATURE_QOS_SCHEDULER_DWRR_CNT . . . . .	541
6.2.2.38	VTSS_FEATURE_QOS_TAG_REMARK_V2 . . . . .	541
6.2.2.39	VTSS_FEATURE_QOS_CLASSIFICATION_V2 . . . . .	542
6.2.2.40	VTSS_FEATURE_QOS_EGRESS_QUEUE_SHAPERS . . . . .	542
6.2.2.41	VTSS_FEATURE_QOS_EGRESS_SHAPERS_DLB . . . . .	542
6.2.2.42	VTSS_FEATURE_QOS_DSCP_CLASS_DP_AWARE . . . . .	542
6.2.2.43	VTSS_FEATURE_QOS_DSCP_REMARK . . . . .	542
6.2.2.44	VTSS_FEATURE_QOS_DSCP_REMARK_V2 . . . . .	543
6.2.2.45	VTSS_FEATURE_QOS_WRED_V3 . . . . .	543
6.2.2.46	VTSS_FEATURE_QOS_POLICER_DLB . . . . .	543
6.2.2.47	VTSS_FEATURE_PACKET . . . . .	543
6.2.2.48	VTSS_FEATURE_PACKET_TX . . . . .	543
6.2.2.49	VTSS_FEATURE_PACKET_RX . . . . .	544
6.2.2.50	VTSS_FEATURE_PACKET_GROUPING . . . . .	544
6.2.2.51	VTSS_FEATURE_PACKET_PORT_REG . . . . .	544
6.2.2.52	VTSS_FEATURE_LAYER2 . . . . .	544

6.2.2.53	VTSS_FEATURE_PVLAN . . . . .	544
6.2.2.54	VTSS_FEATURE_VLAN_PORT_V2 . . . . .	545
6.2.2.55	VTSS_FEATURE_VLAN_TX_TAG . . . . .	545
6.2.2.56	VTSS_FEATURE_MAC_AGE_AUTO . . . . .	545
6.2.2.57	VTSS_FEATURE_MAC_CPU_QUEUE . . . . .	545
6.2.2.58	VTSS_FEATURE_EEE . . . . .	545
6.2.2.59	VTSS_FEATURE_FAN . . . . .	546
6.2.2.60	VTSS_FEATURE_VCAP . . . . .	546
6.2.2.61	VTSS_FEATURE_ACL . . . . .	546
6.2.2.62	VTSS_FEATURE_ACL_V2 . . . . .	546
6.2.2.63	VTSS_FEATURE_VCL . . . . .	546
6.2.2.64	VTSS_FEATURE_TIMESTAMP . . . . .	547
6.2.2.65	VTSS_FEATURE_PHY_TIMESTAMP . . . . .	547
6.2.2.66	VTSS_FEATURE_TIMESTAMP_ONE_STEP . . . . .	547
6.2.2.67	VTSS_FEATURE_TIMESTAMP_LATENCY_COMP . . . . .	547
6.2.2.68	VTSS_FEATURE_TIMESTAMP_ORG_TIME . . . . .	547
6.2.2.69	VTSS_FEATURE_TIMESTAMP_P2P_DELAY_COMP . . . . .	548
6.2.2.70	VTSS_FEATURE_TIMESTAMP_ASYMMETRY_COMP . . . . .	548
6.2.2.71	VTSS_FEATURE_SYNCE . . . . .	548
6.2.2.72	VTSS_FEATURE_NPI . . . . .	548
6.2.2.73	VTSS_FEATURE_LED_POW_REDUC . . . . .	548
6.2.2.74	VTSS_FEATURE_INTERRUPTS . . . . .	549
6.2.2.75	VTSS_FEATURE_IRQ_CONTROL . . . . .	549
6.2.2.76	VTSS_FEATURE_VLAN_TRANSLATION . . . . .	549
6.2.2.77	VTSS_FEATURE_SFLOW . . . . .	549
6.2.2.78	VTSS_FEATURE_SERDES_MACRO_SETTINGS . . . . .	549
6.2.2.79	VTSS_FEATURE_10GBASE_KR . . . . .	550
6.2.2.80	VTSS_FEATURE_10G . . . . .	550
6.2.2.81	VTSS_OPT_VCORE_III . . . . .	550
6.2.2.82	VTSS_FEATURE_FDMA . . . . .	550

6.2.2.83	VTSS_OPT_PCIE_ACCESS	550
6.2.2.84	VTSS_PHY_10G_FIFO_SYNC	551
6.2.2.85	VIPER_B_FIFO_RESET	551
6.2.2.86	VTSS_CHIP CU PHY	551
6.2.2.87	VTSS_OPT_TRACE	551
6.2.2.88	VTSS_OPT_FDMA_IRQ_CONTEXT	551
6.2.2.89	VTSS_OPT_FDMA_DEBUG	552
6.2.2.90	VTSS_OPT_VAUI_EQ_CTRL	552
6.2.2.91	VTSS_PHY_OPT_VERIPHYS	552
6.2.2.92	VTSS_FEATURE_WARM_START [1/2]	552
6.2.2.93	VTSS_FEATURE_SYNCE_10G	552
6.2.2.94	VTSS_FEATURE_EDC_FW_LOAD	553
6.2.2.95	VTSS_FEATURE_WIS	553
6.2.2.96	VTSS_FEATURE_WARM_START [2/2]	553
6.2.2.97	VTSS_ARCH_MALIBU	553
6.2.2.98	VTSS_ARCH_MALIBU_B	553
6.2.2.99	VTSS_ARCH_VENICE_C	554
6.3	vtss_api/include/vtss/api/phy.h File Reference	554
6.3.1	Detailed Description	554
6.3.2	Macro Definition Documentation	554
6.3.2.1	VTSS_PHY_POWER_ACTIPHY_BIT	555
6.3.2.2	VTSS_PHY_POWER_DYNAMIC_BIT	555
6.3.3	Enumeration Type Documentation	555
6.3.3.1	vtss_phy_power_mode_t	555
6.3.3.2	vtss_phy_veriphy_status_t	555
6.4	vtss_api/include/vtss/api/port.h File Reference	556
6.4.1	Detailed Description	558
6.4.2	Macro Definition Documentation	558
6.4.2.1	PORT_CAP_NONE	558
6.4.2.2	PORT_CAP_AUTONEG	558

6.4.2.3	PORT_CAP_10M_HDX . . . . .	559
6.4.2.4	PORT_CAP_10M_FDX . . . . .	559
6.4.2.5	PORT_CAP_100M_HDX . . . . .	559
6.4.2.6	PORT_CAP_100M_FDX . . . . .	559
6.4.2.7	PORT_CAP_1G_FDX . . . . .	559
6.4.2.8	PORT_CAP_2_5G_FDX . . . . .	560
6.4.2.9	PORT_CAP_5G_FDX . . . . .	560
6.4.2.10	PORT_CAP_10G_FDX . . . . .	560
6.4.2.11	PORT_CAP_FLOW_CTRL . . . . .	560
6.4.2.12	PORT_CAP_COPPER . . . . .	560
6.4.2.13	PORT_CAP_FIBER . . . . .	561
6.4.2.14	PORT_CAP_DUAL_COPPER . . . . .	561
6.4.2.15	PORT_CAP_DUAL_FIBER . . . . .	561
6.4.2.16	PORT_CAP_SD_ENABLE . . . . .	561
6.4.2.17	PORT_CAP_SD_HIGH . . . . .	561
6.4.2.18	PORT_CAP_SD_INTERNAL . . . . .	562
6.4.2.19	PORT_CAP_DUAL_FIBER_100FX . . . . .	562
6.4.2.20	PORT_CAP_XAUI_LANE_FLIP . . . . .	562
6.4.2.21	PORT_CAP_VTSS_10G_PHY . . . . .	562
6.4.2.22	PORT_CAP_SFP_DETECT . . . . .	562
6.4.2.23	PORT_CAP_STACKING . . . . .	563
6.4.2.24	PORT_CAP_DUAL_SFP_DETECT . . . . .	563
6.4.2.25	PORT_CAP_SFP_ONLY . . . . .	563
6.4.2.26	PORT_CAP_DUAL_COPPER_100FX . . . . .	563
6.4.2.27	PORT_CAP_HDX . . . . .	563
6.4.2.28	PORT_CAP_TRI_SPEED_FDX . . . . .	564
6.4.2.29	PORT_CAP_TRI_SPEED . . . . .	564
6.4.2.30	PORT_CAP_1G_PHY . . . . .	564
6.4.2.31	PORT_CAP_TRI_SPEED_COPPER . . . . .	564
6.4.2.32	PORT_CAP_TRI_SPEED_FIBER . . . . .	564

6.4.2.33	PORT_CAP_TRI_SPEED_DUAL_COPPER	565
6.4.2.34	PORT_CAP_TRI_SPEED_DUAL_FIBER	565
6.4.2.35	PORT_CAP_TRI_SPEED_DUAL_FIBER_100FX	565
6.4.2.36	PORT_CAP_ANY_FIBER	565
6.4.2.37	PORT_CAP_SPEED_DUAL_ANY_FIBER_FIXED_SPEED	565
6.4.2.38	PORT_CAP_SPEED_DUAL_ANY_FIBER	566
6.4.2.39	PORT_CAP_TRI_SPEED_DUAL_ANY_FIBER	566
6.4.2.40	PORT_CAP_TRI_SPEED_DUAL_ANY_FIBER_FIXED_SFP_SPEED	566
6.4.2.41	PORT_CAP_DUAL_FIBER_1000X	566
6.4.2.42	PORT_CAP_SFP_1G	566
6.4.2.43	PORT_CAP_SFP_2_5G	567
6.4.2.44	PORT_CAP_SFP_SD_HIGH	567
6.4.2.45	PORT_CAP_2_5G_TRI_SPEED_FDX	567
6.4.2.46	PORT_CAP_2_5G_TRI_SPEED	567
6.4.2.47	PORT_CAP_2_5G_TRI_SPEED_COPPER	567
6.4.3	Typedef Documentation	568
6.4.3.1	port_cap_t	568
6.4.4	Enumeration Type Documentation	568
6.4.4.1	vtss_port_speed_t	568
6.4.4.2	vtss_fiber_port_speed_t	568
6.5	vtss_api/include/vtss/api/types.h File Reference	569
6.5.1	Detailed Description	576
6.5.2	Macro Definition Documentation	576
6.5.2.1	PRIu64	576
6.5.2.2	PRIi64	576
6.5.2.3	PRIx64	576
6.5.2.4	VTSS_BIT64	577
6.5.2.5	VTSS_BITMASK64	577
6.5.2.6	VTSS_EXTRACT_BITFIELD64	577
6.5.2.7	VTSS_ENCODE_BITFIELD64	577

6.5.2.8	VTSS_ENCODE_BITMASK64 . . . . .	578
6.5.2.9	TRUE . . . . .	578
6.5.2.10	FALSE . . . . .	578
6.5.2.11	VTSS_PACKET_RATE_DISABLED . . . . .	578
6.5.2.12	VTSS_PORT_COUNT [1/2] . . . . .	578
6.5.2.13	VTSS_PORT_COUNT [2/2] . . . . .	579
6.5.2.14	VTSS_PORTS . . . . .	579
6.5.2.15	VTSS_PORT_NO_NONE . . . . .	579
6.5.2.16	VTSS_PORT_NO_CPU . . . . .	579
6.5.2.17	VTSS_PORT_NO_START . . . . .	579
6.5.2.18	VTSS_PORT_NO_END . . . . .	580
6.5.2.19	VTSS_PORT_ARRAY_SIZE . . . . .	580
6.5.2.20	VTSS_PORT_IS_PORT . . . . .	580
6.5.2.21	VTSS_PRIOS . . . . .	580
6.5.2.22	VTSS_PRIO_NO_NONE . . . . .	580
6.5.2.23	VTSS_PRIO_START . . . . .	581
6.5.2.24	VTSS_PRIO_END . . . . .	581
6.5.2.25	VTSS_PRIO_ARRAY_SIZE . . . . .	581
6.5.2.26	VTSS_QUEUES . . . . .	581
6.5.2.27	VTSS_QUEUE_START . . . . .	581
6.5.2.28	VTSS_QUEUE_END . . . . .	582
6.5.2.29	VTSS_QUEUE_ARRAY_SIZE . . . . .	582
6.5.2.30	VTSS_PCPS . . . . .	582
6.5.2.31	VTSS_PCP_START . . . . .	582
6.5.2.32	VTSS_PCP_END . . . . .	582
6.5.2.33	VTSS_PCP_ARRAY_SIZE . . . . .	583
6.5.2.34	VTSS_DEIS . . . . .	583
6.5.2.35	VTSS_DEI_START . . . . .	583
6.5.2.36	VTSS_DEI_END . . . . .	583
6.5.2.37	VTSS_DEI_ARRAY_SIZE . . . . .	583

6.5.2.38	VTSS_DPLS [1/2]	584
6.5.2.39	VTSS_DPLS [2/2]	584
6.5.2.40	VTSS_DPL_START	584
6.5.2.41	VTSS_DPL_END	584
6.5.2.42	VTSS_DPL_ARRAY_SIZE	584
6.5.2.43	VTSS_BITRATE_DISABLED	585
6.5.2.44	VTSS_VID_NULL	585
6.5.2.45	VTSS_VID_DEFAULT	585
6.5.2.46	VTSS_VID_RESERVED	585
6.5.2.47	VTSS_VIDS	585
6.5.2.48	VTSS_VID_ALL	586
6.5.2.49	VTSSETYPE_VTSS	586
6.5.2.50	VTSS_MAC_ADDR_SZ_BYTES	586
6.5.2.51	MAC_ADDR_BROADCAST	586
6.5.2.52	VTSS_EVCS	586
6.5.2.53	VTSS_ISDX_NONE	587
6.5.2.54	VTSS_AGGRS	587
6.5.2.55	VTSS_AGGR_NO_NONE	587
6.5.2.56	VTSS_AGGR_NO_START	587
6.5.2.57	VTSS_AGGR_NO_END	587
6.5.2.58	VTSS_GLAGS	588
6.5.2.59	VTSS_GLAG_NO_NONE	588
6.5.2.60	VTSS_GLAG_NO_START	588
6.5.2.61	VTSS_GLAG_NO_END	588
6.5.2.62	VTSS_GLAG_PORTS	588
6.5.2.63	VTSS_GLAG_PORT_START	589
6.5.2.64	VTSS_GLAG_PORT_END	589
6.5.2.65	VTSS_GLAG_PORT_ARRAY_SIZE	589
6.5.2.66	VTSS_PACKET_RX_QUEUE_CNT	589
6.5.2.67	VTSS_PACKET_RX_GRP_CNT	589

6.5.2.68	VTSS_PACKET_TX_GRP_CNT . . . . .	590
6.5.2.69	VTSS_PACKET_RX_QUEUE_NONE . . . . .	590
6.5.2.70	VTSS_PACKET_RX_QUEUE_START . . . . .	590
6.5.2.71	VTSS_PACKET_RX_QUEUE_END . . . . .	590
6.5.2.72	VTSS_ACL_POLICERS . . . . .	590
6.5.2.73	VTSS_ACL_POLICER_NO_START . . . . .	591
6.5.2.74	VTSS_ACL_POLICER_NO_END . . . . .	591
6.5.2.75	VTSS_ACL_POLICY_NO_NONE . . . . .	591
6.5.2.76	VTSS_ACL_POLICY_NO_MIN . . . . .	591
6.5.2.77	VTSS_ACL_POLICY_NO_MAX . . . . .	591
6.5.2.78	VTSS_ACL_POLICIES . . . . .	592
6.5.2.79	VTSS_ACL_POLICY_NO_START . . . . .	592
6.5.2.80	VTSS_ACL_POLICY_NO_END . . . . .	592
6.5.2.81	VTSS_HQOS_COUNT . . . . .	592
6.5.2.82	VTSS_HQOS_ID_NONE . . . . .	592
6.5.2.83	VTSS_ONE_MIA . . . . .	593
6.5.2.84	VTSS_ONE_MILL . . . . .	593
6.5.2.85	VTSS_MAX_TIMEINTERVAL . . . . .	593
6.5.2.86	VTSS_INTERVAL_SEC . . . . .	593
6.5.2.87	VTSS_INTERVAL_MS . . . . .	593
6.5.2.88	VTSS_INTERVAL_US . . . . .	594
6.5.2.89	VTSS_INTERVAL_NS . . . . .	594
6.5.2.90	VTSS_INTERVAL_PS . . . . .	594
6.5.2.91	VTSS_SEC_NS_INTERVAL . . . . .	594
6.5.2.92	VTSS_CLOCK_IDENTITY_LENGTH . . . . .	594
6.5.2.93	VTSS_SYNCE_CLK_PORT_ARRAY_SIZE . . . . .	595
6.5.3	Typedef Documentation . . . . .	595
6.5.3.1	i8 . . . . .	595
6.5.3.2	i16 . . . . .	595
6.5.3.3	i32 . . . . .	595

6.5.3.4	i64	595
6.5.3.5	u8	596
6.5.3.6	u16	596
6.5.3.7	u32	596
6.5.3.8	u64	596
6.5.3.9	BOOL	596
6.5.3.10	uintptr_t	597
6.5.3.11	vtss_mac_addr_t	597
6.5.3.12	vtss_isidx_t	597
6.5.3.13	vtss_packet_rx_grp_t	597
6.5.3.14	vtss_packet_tx_grp_t	597
6.5.4	Enumeration Type Documentation	597
6.5.4.1	anonymous enum	597
6.5.4.2	vtss_mem_flags_t	600
6.5.4.3	vtss_port_interface_t	600
6.5.4.4	vtss_serdes_mode_t	601
6.5.4.5	vtss_storm_policer_mode_t	602
6.5.4.6	vtss_policer_type_t	602
6.5.4.7	vtss_vlan_frame_t	602
6.5.4.8	vtss_packet_reg_type_t	603
6.5.4.9	vtss_vdd_t	603
6.5.4.10	vtss_ip_type_t	603
6.5.4.11	vtss_vcap_bit_t	604
6.5.4.12	vtss_vcap_vr_type_t	604
6.5.4.13	vtss_vcap_key_type_t	604
6.5.4.14	vtss_hqos_sch_mode_t	605
6.6	vtss_api/include/vtss_ae_api.h File Reference	605
6.6.1	Detailed Description	605
6.7	vtss_api/include/vtss_afi_api.h File Reference	605
6.7.1	Detailed Description	605

6.8 vtss_api/include/vtss_aneg_api.h File Reference . . . . .	605
6.8.1 Detailed Description . . . . .	606
6.9 vtss_api/include/vtss_api.h File Reference . . . . .	606
6.9.1 Detailed Description . . . . .	606
6.10 vtss_api/include/vtss_evc_api.h File Reference . . . . .	606
6.10.1 Detailed Description . . . . .	607
6.10.2 Function Documentation . . . . .	607
6.10.2.1 vtss_evc_policer_conf_get() . . . . .	607
6.10.2.2 vtss_evc_policer_conf_set() . . . . .	607
6.11 vtss_api/include/vtss_fdma_api.h File Reference . . . . .	608
6.11.1 Detailed Description . . . . .	609
6.11.2 Macro Definition Documentation . . . . .	609
6.11.2.1 VTSS_FDMA_CH_CNT . . . . .	610
6.11.2.2 VTSS_FDMA_DCACHE_LINE_SIZE_BYTES . . . . .	610
6.11.2.3 VTSS_FDMA_HDR_SIZE_BYTES . . . . .	610
6.11.2.4 VTSS_FDMA_MAX_DATA_PER_DCACHE_LINE_SIZE_BYTES . . . . .	610
6.11.2.5 VTSS_FDMA_MIN_FRAME_SIZE_BYTES . . . . .	610
6.11.2.6 VTSS_FDMA_MIN_DATA_PER_INJ_SOF_DCACHE_LINE_SIZE_BYTES . . . . .	611
6.11.2.7 VTSS_FDMA_MIN_DATA_PER_XTR_SOF_DCACHE_LINE_SIZE_BYTES . . . . .	611
6.11.2.8 VTSS_FDMA_MIN_DATA_PER_NON_SOF_DCACHE_LINE_SIZE_BYTES . . . . .	611
6.11.2.9 VTSS_FDMA_MAX_FRAME_SIZE_BYTES . . . . .	611
6.11.2.10 VTSS_OS_DCACHE_LINE_SIZE_BYTES . . . . .	611
6.11.2.11 VTSS_OS_COMPILER_ATTRIBUTE_ALIGNED . . . . .	612
6.11.2.12 VTSS_AFI_FPS_MAX . . . . .	612
6.11.3 Typedef Documentation . . . . .	612
6.11.3.1 vtss_fdma_ch_t . . . . .	612
6.11.3.2 vtss_fdma_list_t . . . . .	614
6.11.4 Enumeration Type Documentation . . . . .	614
6.11.4.1 vtss_fdma_ch_usage_t . . . . .	614
6.11.4.2 vtss_fdma_dcba_type_t . . . . .	615

6.11.5 Function Documentation . . . . .	615
6.11.5.1 vtss_fdma_uninit() . . . . .	615
6.11.5.2 vtss_fdma_cfg() . . . . .	616
6.11.5.3 vtss_fdma_dcb_release() . . . . .	616
6.11.5.4 vtss_fdma_tx() . . . . .	617
6.11.5.5 vtss_fdma_tx_info_init() . . . . .	618
6.11.5.6 vtss_fdma_dcb_get() . . . . .	618
6.11.5.7 vtss_fdma_throttle_cfg_get() . . . . .	619
6.11.5.8 vtss_fdma_throttle_cfg_set() . . . . .	619
6.11.5.9 vtss_fdma_throttle_tick() . . . . .	620
6.11.5.10 vtss_fdma_stats_clr() . . . . .	620
6.11.5.11 vtss_fdma_irq_handler() . . . . .	621
6.12 vtss_api/include/vtss_gfp_api.h File Reference . . . . .	622
6.12.1 Detailed Description . . . . .	622
6.13 vtss_api/include/vtss_hqos_api.h File Reference . . . . .	622
6.13.1 Detailed Description . . . . .	623
6.14 vtss_api/include/vtss_i2c_api.h File Reference . . . . .	623
6.14.1 Detailed Description . . . . .	623
6.15 vtss_api/include/vtss_init_api.h File Reference . . . . .	623
6.15.1 Detailed Description . . . . .	625
6.15.2 Macro Definition Documentation . . . . .	626
6.15.2.1 VTSS_I2C_NO_MULTIPLEXER . . . . .	626
6.15.3 Typedef Documentation . . . . .	626
6.15.3.1 vtss_reg_read_t . . . . .	626
6.15.3.2 vtss_reg_write_t . . . . .	626
6.15.3.3 vtss_i2c_read_t . . . . .	627
6.15.3.4 vtss_i2c_write_t . . . . .	627
6.15.3.5 vtss_spi_read_write_t . . . . .	628
6.15.3.6 vtss_spi_32bit_read_write_t . . . . .	628
6.15.3.7 vtss_spi_64bit_read_write_t . . . . .	629

6.15.3.8 <code>vtss_miim_read_t</code>	629
6.15.3.9 <code>vtss_miim_write_t</code>	630
6.15.3.10 <code>vtss_mmd_read_t</code>	630
6.15.3.11 <code>vtss_mmd_read_inc_t</code>	631
6.15.3.12 <code>vtss_mmd_write_t</code>	631
6.15.4 Enumeration Type Documentation	632
6.15.4.1 <code>vtss_target_type_t</code>	632
6.15.4.2 <code>vtss_port_mux_mode_t</code>	633
6.15.4.3 <code>vtss_restart_t</code>	633
6.15.5 Function Documentation	633
6.15.5.1 <code>vtss_inst_get()</code>	633
6.15.5.2 <code>vtss_inst_create()</code>	634
6.15.5.3 <code>vtss_inst_destroy()</code>	634
6.15.5.4 <code>vtss_init_conf_get()</code>	635
6.15.5.5 <code>vtss_init_conf_set()</code>	635
6.15.5.6 <code>vtss_restart_conf_end()</code>	635
6.15.5.7 <code>vtss_restart_status_get()</code>	636
6.15.5.8 <code>vtss_restart_conf_get()</code>	636
6.15.5.9 <code>vtss_restart_conf_set()</code>	636
6.16 <code>vtss_api/include/vtss_l2_api.h</code> File Reference	637
6.16.1 Detailed Description	645
6.16.2 Macro Definition Documentation	645
6.16.2.1 <code>VTSS_MAC_ADDRS</code>	645
6.16.2.2 <code>VTSS_VSTAX_UPSIDS</code>	646
6.16.2.3 <code>VTSS_VSTAX_UPSID_START</code>	646
6.16.2.4 <code>VTSS_VSTAX_UPSID_MIN</code>	646
6.16.2.5 <code>VTSS_VSTAX_UPSID_MAX</code>	646
6.16.2.6 <code>VTSS_VSTAX_UPSID_LEGAL</code>	646
6.16.2.7 <code>VTSS_VSTAX_UPSID_UNDEF</code>	647
6.16.2.8 <code>VTSS_UPSPN_CPU</code>	647

6.16.2.9 VTSS_UPSPN_NONE . . . . .	647
6.16.2.10 VTSS_MSTIS . . . . .	647
6.16.2.11 VTSS_MSTI_START . . . . .	647
6.16.2.12 VTSS_MSTI_END . . . . .	648
6.16.2.13 VTSS_MSTI_ARRAY_SIZE . . . . .	648
6.16.2.14 VTSS_VCL_IDS . . . . .	648
6.16.2.15 VTSS_VCL_ID_START . . . . .	648
6.16.2.16 VTSS_VCL_ID_END . . . . .	648
6.16.2.17 VTSS_VCL_ARRAY_SIZE . . . . .	649
6.16.2.18 VTSS_VCE_ID_LAST . . . . .	649
6.16.2.19 VTSS_VLAN_TRANS_GROUP_MAX_CNT . . . . .	649
6.16.2.20 VTSS_VLAN_TRANS_MAX_CNT . . . . .	649
6.16.2.21 VTSS_VLAN_TRANS_NULL_GROUP_ID . . . . .	649
6.16.2.22 VTSS_VLAN_TRANS_FIRST_GROUP_ID . . . . .	650
6.16.2.23 VTSS_VLAN_TRANS_VID_START . . . . .	650
6.16.2.24 VTSS_VLAN_TRANS_MAX_VLAN_ID . . . . .	650
6.16.2.25 VTSS_VLAN_TRANS_LAST_GROUP_ID . . . . .	650
6.16.2.26 VTSS_VLAN_TRANS_VALID_GROUP_CHECK . . . . .	650
6.16.2.27 VTSS_VLAN_TRANS_VALID_VLAN_CHECK . . . . .	651
6.16.2.28 VTSS_VLAN_TRANS_NULL_CHECK . . . . .	651
6.16.2.29 VTSS_VLAN_TRANS_PORT_BF_SIZE . . . . .	651
6.16.2.30 VTSS_PVLANS . . . . .	651
6.16.2.31 VTSS_PVLAN_NO_START . . . . .	652
6.16.2.32 VTSS_PVLAN_NO_END . . . . .	652
6.16.2.33 VTSS_PVLAN_ARRAY_SIZE . . . . .	652
6.16.2.34 VTSS_PVLAN_NO_DEFAULT . . . . .	652
6.16.2.35 VTSS_ERPIS . . . . .	652
6.16.2.36 VTSS_ERPI_START . . . . .	653
6.16.2.37 VTSS_ERPI_END . . . . .	653
6.16.2.38 VTSS_ERPI_ARRAY_SIZE . . . . .	653

6.16.3	Typedef Documentation . . . . .	653
6.16.3.1	vtss_vt_id_t . . . . .	653
6.16.4	Enumeration Type Documentation . . . . .	653
6.16.4.1	vtss_stp_state_t . . . . .	653
6.16.4.2	vtss_vlan_port_type_t . . . . .	654
6.16.4.3	vtss_vlan_tx_tag_t . . . . .	654
6.16.4.4	vtss_vce_type_t . . . . .	654
6.16.4.5	vtss_mirror_tag_t . . . . .	655
6.16.4.6	vtss_eps_port_type_t . . . . .	655
6.16.4.7	vtss_eps_selector_t . . . . .	655
6.16.4.8	vtss_erps_state_t . . . . .	656
6.16.4.9	vtss_vstax_topology_type_t . . . . .	656
6.16.5	Function Documentation . . . . .	656
6.16.5.1	vtss_mac_table_add() . . . . .	656
6.16.5.2	vtss_mac_table_del() . . . . .	658
6.16.5.3	vtss_mac_table_get() . . . . .	658
6.16.5.4	vtss_mac_table_get_next() . . . . .	659
6.16.5.5	vtss_mac_table_age_time_get() . . . . .	659
6.16.5.6	vtss_mac_table_age_time_set() . . . . .	659
6.16.5.7	vtss_mac_table_age() . . . . .	660
6.16.5.8	vtss_mac_table_vlan_age() . . . . .	660
6.16.5.9	vtss_mac_table_flush() . . . . .	661
6.16.5.10	vtss_mac_table_port_flush() . . . . .	661
6.16.5.11	vtss_mac_table_vlan_flush() . . . . .	661
6.16.5.12	vtss_mac_table_vlan_port_flush() . . . . .	662
6.16.5.13	vtss_mac_table_upsid_flush() . . . . .	662
6.16.5.14	vtss_mac_table_upsid_upspn_flush() . . . . .	662
6.16.5.15	vtss_mac_table_glag_add() . . . . .	663
6.16.5.16	vtss_mac_table_glag_flush() . . . . .	663
6.16.5.17	vtss_mac_table_vlan_glag_flush() . . . . .	664

6.16.5.18 vtss_mac_table_status_get()	664
6.16.5.19 vtss_learn_port_mode_get()	664
6.16.5.20 vtss_learn_port_mode_set()	665
6.16.5.21 vtss_port_state_get()	665
6.16.5.22 vtss_port_state_set()	666
6.16.5.23 vtss_stp_port_state_get()	666
6.16.5.24 vtss_stp_port_state_set()	667
6.16.5.25 vtss_mstp_vlan_msti_get()	667
6.16.5.26 vtss_mstp_vlan_msti_set()	667
6.16.5.27 vtss_mstp_port_msti_state_get()	668
6.16.5.28 vtss_mstp_port_msti_state_set()	668
6.16.5.29 vtss_vlan_conf_get()	669
6.16.5.30 vtss_vlan_conf_set()	669
6.16.5.31 vtss_vlan_port_conf_get()	669
6.16.5.32 vtss_vlan_port_conf_set()	671
6.16.5.33 vtss_vlan_port_members_get()	671
6.16.5.34 vtss_vlan_port_members_set()	672
6.16.5.35 vtss_vlan_vid_conf_get()	672
6.16.5.36 vtss_vlan_vid_conf_set()	673
6.16.5.37 vtss_vlan_tx_tag_get()	673
6.16.5.38 vtss_vlan_tx_tag_set()	673
6.16.5.39 vtss_vcl_port_conf_get()	674
6.16.5.40 vtss_vcl_port_conf_set()	674
6.16.5.41 vtss_vce_init()	675
6.16.5.42 vtss_vce_add()	675
6.16.5.43 vtss_vce_del()	676
6.16.5.44 vtss_vlan_trans_group_add()	676
6.16.5.45 vtss_vlan_trans_group_del()	676
6.16.5.46 vtss_vlan_trans_group_get()	677
6.16.5.47 vtss_vlan_trans_group_to_port_set()	677

6.16.5.48 vtss_vlan_trans_group_to_port_get()	678
6.16.5.49 vtss_isolated_vlan_get()	678
6.16.5.50 vtss_isolated_vlan_set()	679
6.16.5.51 vtss_isolated_port_members_get()	679
6.16.5.52 vtss_isolated_port_members_set()	679
6.16.5.53 vtss_pvlan_port_members_get()	680
6.16.5.54 vtss_pvlan_port_members_set()	680
6.16.5.55 vtss_apvlan_port_members_get()	681
6.16.5.56 vtss_apvlan_port_members_set()	681
6.16.5.57 vtss_dgroup_port_conf_get()	682
6.16.5.58 vtss_dgroup_port_conf_set()	682
6.16.5.59 vtss_sflow_port_conf_get()	682
6.16.5.60 vtss_sflow_port_conf_set()	683
6.16.5.61 vtss_sflow_sampling_rate_convert()	683
6.16.5.62 vtss_aggr_port_members_get()	684
6.16.5.63 vtss_aggr_port_members_set()	684
6.16.5.64 vtss_aggr_mode_get()	685
6.16.5.65 vtss_aggr_mode_set()	685
6.16.5.66 vtss_aggr_glag_members_get()	685
6.16.5.67 vtss_vstax_glag_get()	687
6.16.5.68 vtss_vstax_glag_set()	687
6.16.5.69 vtss_mirror_conf_get()	688
6.16.5.70 vtss_mirror_conf_set()	688
6.16.5.71 vtss_mirror_monitor_port_get()	688
6.16.5.72 vtss_mirror_monitor_port_set()	689
6.16.5.73 vtss_mirror_ingress_ports_get()	689
6.16.5.74 vtss_mirror_ingress_ports_set()	690
6.16.5.75 vtss_mirror_egress_ports_get()	690
6.16.5.76 vtss_mirror_egress_ports_set()	690
6.16.5.77 vtss_mirror_cpu_ingress_get()	691

6.16.5.78 vtss_mirror_cpu_ingress_set()	691
6.16.5.79 vtss_mirror_cpu_egress_get()	691
6.16.5.80 vtss_mirror_cpu_egress_set()	692
6.16.5.81 vtss_uc_flood_members_get()	692
6.16.5.82 vtss_uc_flood_members_set()	693
6.16.5.83 vtss_mc_flood_members_get()	693
6.16.5.84 vtss_mc_flood_members_set()	693
6.16.5.85 vtss_ipv4_mc_flood_members_get()	694
6.16.5.86 vtss_ipv4_mc_flood_members_set()	694
6.16.5.87 vtss_ipv6_mc_flood_members_get()	695
6.16.5.88 vtss_ipv6_mc_flood_members_set()	695
6.16.5.89 vtss_ipv6_mc_ctrl_flood_get()	695
6.16.5.90 vtss_ipv6_mc_ctrl_flood_set()	696
6.16.5.91 vtss_eps_port_conf_get()	696
6.16.5.92 vtss_eps_port_conf_set()	697
6.16.5.93 vtss_eps_port_selector_get()	697
6.16.5.94 vtss_eps_port_selector_set()	697
6.16.5.95 vtss_erps_vlan_member_get()	698
6.16.5.96 vtss_erps_vlan_member_set()	698
6.16.5.97 vtss_erps_port_state_get()	699
6.16.5.98 vtss_erps_port_state_set()	699
6.16.5.99 vtss_vstax_conf_get()	700
6.16.5.100 vtss_vstax_conf_set()	700
6.16.5.101 vtss_vstax_port_conf_get()	700
6.16.5.102 vtss_vstax_port_conf_set()	701
6.16.5.103 vtss_vstax_master_upsid_get()	701
6.16.5.104 vtss_vstax_master_upsid_set()	702
6.16.5.105 vtss_vstax_topology_set()	702
6.17 vtss_api/include/vtss_l3_api.h File Reference	702
6.17.1 Detailed Description	703

6.18 vtss_api/include/vtss_mac10g_api.h File Reference . . . . .	703
6.18.1 Detailed Description . . . . .	703
6.19 vtss_api/include/vtss_misc_api.h File Reference . . . . .	703
6.19.1 Detailed Description . . . . .	709
6.19.2 Macro Definition Documentation . . . . .	709
6.19.2.1 VTSS_OS_TIMESTAMP_TYPE . . . . .	709
6.19.2.2 VTSS_OS_TIMESTAMP . . . . .	709
6.19.3 Typedef Documentation . . . . .	709
6.19.3.1 tod_get_ns_cnt_cb_t . . . . .	709
6.19.4 Enumeration Type Documentation . . . . .	710
6.19.4.1 vtss_trace_layer_t . . . . .	710
6.19.4.2 vtss_trace_group_t . . . . .	710
6.19.4.3 vtss_trace_level_t . . . . .	711
6.19.4.4 vtss_debug_layer_t . . . . .	711
6.19.4.5 vtss_debug_group_t . . . . .	711
6.19.4.6 vtss_gpio_mode_t . . . . .	712
6.19.4.7 vtss_sgpio_mode_t . . . . .	713
6.19.4.8 vtss_sgpio_bmode_t . . . . .	713
6.19.4.9 vtss_irq_t . . . . .	714
6.19.4.10 vtss_eee_state_select_t . . . . .	714
6.19.5 Function Documentation . . . . .	714
6.19.5.1 vtss_trace_conf_get() . . . . .	715
6.19.5.2 vtss_trace_conf_set() . . . . .	715
6.19.5.3 vtss_callout_trace_printf() . . . . .	715
6.19.5.4 vtss_callout_trace_hex_dump() . . . . .	716
6.19.5.5 vtss_debug_info_get() . . . . .	717
6.19.5.6 vtss_debug_info_print() . . . . .	717
6.19.5.7 vtss_callout_lock() . . . . .	717
6.19.5.8 vtss_callout_unlock() . . . . .	718
6.19.5.9 vtss_debug_lock() . . . . .	718

6.19.5.10 vtss_debug_unlock()	718
6.19.5.11 vtss_reg_read()	719
6.19.5.12 vtss_reg_write()	719
6.19.5.13 vtss_reg_write_masked()	720
6.19.5.14 vtss_intr_sticky_clear()	720
6.19.5.15 vtss_chip_id_get()	720
6.19.5.16 vtss_poll_1sec()	721
6.19.5.17 vtss_ptp_event_poll()	721
6.19.5.18 vtss_ptp_event_enable()	722
6.19.5.19 vtss_dev_all_event_poll()	722
6.19.5.20 vtss_dev_all_event_enable()	723
6.19.5.21 vtss_gpio_mode_set()	723
6.19.5.22 vtss_gpio_direction_set()	724
6.19.5.23 vtss_gpio_read()	724
6.19.5.24 vtss_gpio_write()	724
6.19.5.25 vtss_gpio_event_poll()	725
6.19.5.26 vtss_gpio_event_enable()	725
6.19.5.27 vtss_sgpio_conf_get()	726
6.19.5.28 vtss_sgpio_conf_set()	726
6.19.5.29 vtss_sgpio_read()	727
6.19.5.30 vtss_sgpio_event_poll()	727
6.19.5.31 vtss_sgpio_event_enable()	728
6.19.5.32 vtss_intr_cfg()	728
6.19.5.33 vtss_intr_mask_set()	729
6.19.5.34 vtss_intr_status_get()	729
6.19.5.35 vtss_intr_pol_negation()	729
6.19.5.36 vtss_irq_conf_get()	730
6.19.5.37 vtss_irq_conf_set()	730
6.19.5.38 vtss_irq_status_get_and_mask()	731
6.19.5.39 vtss_irq_enable()	731

6.19.5.40 <a href="#">vtss_tod_get_ns_cnt()</a>	731
6.19.5.41 <a href="#">vtss_tod_set_ns_cnt_cb()</a>	732
6.19.5.42 <a href="#">vtss_temp_sensor_init()</a>	732
6.19.5.43 <a href="#">vtss_temp_sensor_get()</a>	732
6.19.5.44 <a href="#">vtss_fan_rotation_get()</a>	733
6.19.5.45 <a href="#">vtss_fan_cool_lvl_set()</a>	733
6.19.5.46 <a href="#">vtss_fan_controller_init()</a>	734
6.19.5.47 <a href="#">vtss_fan_cool_lvl_get()</a>	734
6.19.5.48 <a href="#">vtss_eee_port_conf_set()</a>	734
6.19.5.49 <a href="#">vtss_eee_port_state_set()</a>	735
6.19.5.50 <a href="#">vtss_eee_port_counter_get()</a>	735
6.19.5.51 <a href="#">vtss_debug_reg_check_set()</a>	736
6.20 <a href="#">vtss_api/include/vtss_mpls_api.h</a> File Reference	736
6.20.1 Detailed Description	736
6.21 <a href="#">vtss_api/include/vtss_oam_api.h</a> File Reference	737
6.21.1 Detailed Description	737
6.22 <a href="#">vtss_api/include/vtss_oha_api.h</a> File Reference	737
6.22.1 Detailed Description	737
6.23 <a href="#">vtss_api/include/vtss_os.h</a> File Reference	737
6.23.1 Detailed Description	737
6.24 <a href="#">vtss_api/include/vtss_os_custom.h</a> File Reference	737
6.24.1 Detailed Description	738
6.24.2 Macro Definition Documentation	738
6.24.2.1 <a href="#">uint</a>	738
6.24.2.2 <a href="#">ulong</a>	738
6.24.2.3 <a href="#">VTSS_MSLEEP</a>	739
6.24.2.4 <a href="#">VTSS_MTIMER_START</a>	739
6.24.2.5 <a href="#">VTSS_MTIMER_TIMEOUT</a>	739
6.24.2.6 <a href="#">VTSS_MTIMER_CANCEL</a>	739
6.24.2.7 <a href="#">VTSS_DIV64</a>	739

6.24.2.8 VTSS_MOD64 . . . . .	740
6.24.2.9 VTSS_LABS . . . . .	740
6.24.2.10 VTSS_LLABS . . . . .	740
6.24.2.11 VTSS_OS_CTZ . . . . .	740
6.24.2.12 VTSS_OS_CTZ64 . . . . .	741
6.24.2.13 VTSS_OS_MALLOC . . . . .	741
6.24.2.14 VTSS_OS_FREE . . . . .	741
6.24.2.15 VTSS_OS_RAND . . . . .	741
6.24.3 Typedef Documentation . . . . .	742
6.24.3.1 vtss_mtimer_t . . . . .	742
6.25 vtss_api/include/vtss_os_ecos.h File Reference . . . . .	742
6.25.1 Detailed Description . . . . .	743
6.25.2 Macro Definition Documentation . . . . .	743
6.25.2.1 VTSS_MSLEEP . . . . .	743
6.25.2.2 VTSS_NSLEEP . . . . .	744
6.25.2.3 VTSS_MTIMER_START . . . . .	744
6.25.2.4 VTSS_MTIMER_TIMEOUT . . . . .	744
6.25.2.5 VTSS_MTIMER_CANCEL . . . . .	744
6.25.2.6 VTSS_TIME_OF_DAY . . . . .	744
6.25.2.7 VTSS_DIV64 . . . . .	745
6.25.2.8 VTSS_MOD64 . . . . .	745
6.25.2.9 VTSS_LABS . . . . .	745
6.25.2.10 VTSS_LLABS . . . . .	745
6.25.2.11 VTSS_OS_CTZ . . . . .	746
6.25.2.12 VTSS_OS_CTZ64 . . . . .	746
6.25.2.13 VTSS_OS_MALLOC . . . . .	746
6.25.2.14 VTSS_OS_FREE . . . . .	747
6.25.2.15 VTSS_OS_RAND . . . . .	747
6.25.2.16 VTSS_OS_REORDER_BARRIER . . . . .	747
6.25.2.17 VTSS_OS_COMPILER_ATTRIBUTE_ALIGNED . . . . .	747

6.25.2.18 VTSS_OS_DCACHE_LINE_SIZE_BYTES . . . . .	748
6.25.2.19 VTSS_OS_DCACHE_INVALIDATE . . . . .	748
6.25.2.20 VTSS_OS_DCACHE_FLUSH . . . . .	748
6.25.2.21 VTSS_OS_VIRT_TO_PHYS . . . . .	748
6.25.2.22 VTSS_OS_NTOHL . . . . .	749
6.25.2.23 VTSS_OS_SCHEDULER_FLAGS . . . . .	749
6.25.2.24 VTSS_OS_SCHEDULER_LOCK . . . . .	749
6.25.2.25 VTSS_OS_SCHEDULER_UNLOCK . . . . .	749
6.25.2.26 VTSS_OS_INTERRUPT_FLAGS . . . . .	750
6.25.2.27 VTSS_OS_INTERRUPT_DISABLE . . . . .	750
6.25.2.28 VTSS_OS_INTERRUPT_RESTORE . . . . .	750
6.25.3 TYPEDOC Documentation . . . . .	750
6.25.3.1 vtss_mtimer_t . . . . .	750
6.25.4 FUNCTION Documentation . . . . .	750
6.25.4.1 llabs() . . . . .	750
6.25.4.2 vtss_callout_malloc() . . . . .	751
6.25.4.3 vtss_callout_free() . . . . .	751
6.26 vtss_api/include/vtss_os_linux.h File Reference . . . . .	752
6.26.1 Detailed Description . . . . .	752
6.26.2 MACRO Definition Documentation . . . . .	753
6.26.2.1 VTSS_OS_BIG_ENDIAN . . . . .	753
6.26.2.2 VTSS_OS_NTOHL . . . . .	753
6.26.2.3 VTSS_NSLEEP . . . . .	753
6.26.2.4 VTSS_MSLEEP . . . . .	754
6.26.2.5 VTSS_MTIMER_START . . . . .	754
6.26.2.6 VTSS_MTIMER_TIMEOUT . . . . .	754
6.26.2.7 VTSS_MTIMER_CANCEL . . . . .	755
6.26.2.8 VTSS_TIME_OF_DAY . . . . .	755
6.26.2.9 VTSS_OS_SCHEDULER_FLAGS . . . . .	755
6.26.2.10 VTSS_OS_SCHEDULER_LOCK . . . . .	755

6.26.2.11 VTSS_OS_SCHEDULER_UNLOCK . . . . .	756
6.26.2.12 VTSS_DIV64 . . . . .	756
6.26.2.13 VTSS_MOD64 . . . . .	756
6.26.2.14 VTSS_LABS . . . . .	756
6.26.2.15 VTSS_LLabs . . . . .	757
6.26.2.16 VTSS_OS_CTZ . . . . .	757
6.26.2.17 VTSS_OS_CTZ64 . . . . .	758
6.26.2.18 VTSS_OS_MALLOC . . . . .	758
6.26.2.19 VTSS_OS_FREE . . . . .	759
6.26.2.20 VTSS_OS_RAND . . . . .	759
6.27 vtss_api/include/vtss_otn_api.h File Reference . . . . .	759
6.27.1 Detailed Description . . . . .	759
6.28 vtss_api/include/vtss_packet_api.h File Reference . . . . .	759
6.28.1 Detailed Description . . . . .	763
6.28.2 Macro Definition Documentation . . . . .	763
6.28.2.1 VTSS_PRIO_SUPER . . . . .	763
6.28.2.2 VTSS_VSTAX_TTL_PORT . . . . .	763
6.28.2.3 VTSS_VSTAX_HDR_SIZE . . . . .	763
6.28.2.4 VTSS_JR1_PACKET_HDR_SIZE_BYTES . . . . .	764
6.28.2.5 VTSS_JR2_PACKET_HDR_SIZE_BYTES . . . . .	764
6.28.2.6 VTSS_SVL_PACKET_HDR_SIZE_BYTES . . . . .	764
6.28.2.7 VTSS_L26_PACKET_HDR_SIZE_BYTES . . . . .	764
6.28.2.8 VTSS_PACKET_HDR_SIZE_BYTES . . . . .	764
6.28.2.9 VTSS_SVL_RX_IFH_SIZE . . . . .	765
6.28.2.10 VTSS_JR1_RX_IFH_SIZE . . . . .	765
6.28.2.11 VTSS_L26_RX_IFH_SIZE . . . . .	765
6.28.2.12 VTSS_JR2_RX_IFH_SIZE . . . . .	765
6.28.2.13 VTSS_PACKET_TX_IFH_MAX . . . . .	765
6.28.3 Enumeration Type Documentation . . . . .	765
6.28.3.1 vtss_packet_filter_t . . . . .	765

6.28.3.2 <code>vtss_vstax_fwd_mode_t</code>	766
6.28.3.3 <code>vtss_packet_oam_type_t</code>	766
6.28.3.4 <code>vtss_packet_ptp_action_t</code>	767
6.28.3.5 <code>vtss_tag_type_t</code>	767
6.28.3.6 <code>vtss_packet_rx_hints_t</code>	767
6.28.3.7 <code>vtss_packet_tx_vstax_t</code>	768
6.28.4 Function Documentation	769
6.28.4.1 <code>vtss_npi_conf_get()</code>	769
6.28.4.2 <code>vtss_npi_conf_set()</code>	769
6.28.4.3 <code>vtss_packet_rx_conf_get()</code>	770
6.28.4.4 <code>vtss_packet_rx_conf_set()</code>	770
6.28.4.5 <code>vtss_packet_rx_port_conf_get()</code>	770
6.28.4.6 <code>vtss_packet_rx_port_conf_set()</code>	772
6.28.4.7 <code>vtss_packet_rx_frame_get()</code>	772
6.28.4.8 <code>vtss_packet_rx_frame_get_raw()</code>	773
6.28.4.9 <code>vtss_packet_rx_frame_discard()</code>	773
6.28.4.10 <code>vtss_packet_tx_frame_port()</code>	774
6.28.4.11 <code>vtss_packet_tx_frame_port_vlan()</code>	774
6.28.4.12 <code>vtss_packet_tx_frame_vlan()</code>	775
6.28.4.13 <code>vtss_packet_frame_info_init()</code>	775
6.28.4.14 <code>vtss_packet_frame_filter()</code>	775
6.28.4.15 <code>vtss_packet_port_info_init()</code>	776
6.28.4.16 <code>vtss_packet_port_filter_get()</code>	776
6.28.4.17 <code>vtss_packet_tx_frame_vstax()</code>	777
6.28.4.18 <code>vtss_packet_vstax_header2frame()</code>	777
6.28.4.19 <code>vtss_packet_vstax_frame2header()</code>	778
6.28.4.20 <code>vtss_packet_rx_hdr_decode()</code>	778
6.28.4.21 <code>vtss_packet_tx_hdr_encode()</code>	779
6.28.4.22 <code>vtss_packet_tx_hdr_compile()</code>	779
6.28.4.23 <code>vtss_packet_tx_frame()</code>	780

6.28.4.24 <code>vtss_packet_tx_info_init()</code>	780
6.28.4.25 <code>vtss_packet_dma_conf_get()</code>	781
6.28.4.26 <code>vtss_packet_dma_conf_set()</code>	781
6.28.4.27 <code>vtss_packet_dma_offset()</code>	782
6.29 <code>vtss_api/include/vtss_pcs_10gbase_r_api.h</code> File Reference	782
6.29.1 Detailed Description	782
6.30 <code>vtss_api/include/vtss_phy_10g_api.h</code> File Reference	782
6.30.1 Detailed Description	796
6.30.2 Macro Definition Documentation	796
6.30.2.1 <code>VTSS_SLEWRATE_25PS</code>	797
6.30.2.2 <code>VTSS_SLEWRATE_35PS</code>	797
6.30.2.3 <code>VTSS_SLEWRATE_55PS</code>	797
6.30.2.4 <code>VTSS_SLEWRATE_70PS</code>	797
6.30.2.5 <code>VTSS_SLEWRATE_120PS</code>	797
6.30.2.6 <code>VTSS_SLEWRATE_INVALID</code>	798
6.30.2.7 <code>BOOLEAN_STORAGE_COUNT</code>	798
6.30.2.8 <code>UNSIGNED_STORAGE_COUNT</code>	798
6.30.2.9 <code>PHASE_POINTS</code>	798
6.30.2.10 <code>AMPLITUDE_POINTS</code>	798
6.30.2.11 <code>VTSS_PHY_10G_ONE_LINE_ACTIVE</code>	799
6.30.2.12 <code>VTSS_PHY_10G_MACSEC_DISABLED</code>	799
6.30.2.13 <code>VTSS_PHY_10G_TIMESTAMP_DISABLED</code>	799
6.30.2.14 <code>VTSS_PHY_10G_MACSEC_KEY_128</code>	799
6.30.2.15 <code>VTSS_10G_PHY_GPIO_MAX</code>	799
6.30.2.16 <code>VTSS_10G_PHY_GPIO_MAL_MAX</code>	800
6.30.2.17 <code>VTSS_PHY_10G_LINK_LOS_EV</code>	800
6.30.2.18 <code>VTSS_PHY_10G_RX_LOL_EV</code> [1/2]	800
6.30.2.19 <code>VTSS_PHY_10G_TX_LOL_EV</code> [1/2]	800
6.30.2.20 <code>VTSS_PHY_10G_LOPC_EV</code>	800
6.30.2.21 <code>VTSS_PHY_10G_HIGH_BER_EV</code>	801

6.30.2.22 VTSS_PHY_10G_MODULE_STAT_EV . . . . .	801
6.30.2.23 VTSS_PHY_10G_PCS_RECEIVE_FAULT_EV . . . . .	801
6.30.2.24 VTSS_PHY_EWIS_SEF_EV . . . . .	801
6.30.2.25 VTSS_PHY_EWIS_FPLM_EV . . . . .	801
6.30.2.26 VTSS_PHY_EWIS_FAIS_EV . . . . .	802
6.30.2.27 VTSS_PHY_EWIS_LOF_EV . . . . .	802
6.30.2.28 VTSS_PHY_EWIS_RDIL_EV . . . . .	802
6.30.2.29 VTSS_PHY_EWIS_AISL_EV . . . . .	802
6.30.2.30 VTSS_PHY_EWIS_LCDP_EV . . . . .	802
6.30.2.31 VTSS_PHY_EWIS_PLMP_EV . . . . .	803
6.30.2.32 VTSS_PHY_EWIS_AISP_EV . . . . .	803
6.30.2.33 VTSS_PHY_EWIS_LOPP_EV . . . . .	803
6.30.2.34 VTSS_PHY_EWIS_UNEQP_EV . . . . .	803
6.30.2.35 VTSS_PHY_EWIS_FEUNEQP_EV . . . . .	803
6.30.2.36 VTSS_PHY_EWIS_FERDIP_EV . . . . .	804
6.30.2.37 VTSS_PHY_EWIS_REIL_EV . . . . .	804
6.30.2.38 VTSS_PHY_EWIS_REIP_EV . . . . .	804
6.30.2.39 VTSS_PHY_EWIS_B1_NZ_EV . . . . .	804
6.30.2.40 VTSS_PHY_EWIS_B2_NZ_EV . . . . .	804
6.30.2.41 VTSS_PHY_EWIS_B3_NZ_EV . . . . .	805
6.30.2.42 VTSS_PHY_EWIS_REIL_NZ_EV . . . . .	805
6.30.2.43 VTSS_PHY_EWIS_REIP_NZ_EV . . . . .	805
6.30.2.44 VTSS_PHY_EWIS_B1_THRESH_EV . . . . .	805
6.30.2.45 VTSS_PHY_EWIS_B2_THRESH_EV . . . . .	805
6.30.2.46 VTSS_PHY_EWIS_B3_THRESH_EV . . . . .	806
6.30.2.47 VTSS_PHY_EWIS_REIL_THRESH_EV . . . . .	806
6.30.2.48 VTSS_PHY_EWIS_REIP_THRESH_EV . . . . .	806
6.30.2.49 VTSS_PHY_10G_RX_LOS_EV . . . . .	806
6.30.2.50 VTSS_PHY_10G_RX_LOL_EV [2/2] . . . . .	806
6.30.2.51 VTSS_PHY_10G_TX_LOL_EV [2/2] . . . . .	807

6.30.2.52 VTSS_PHY_10G_RX_CHAR_DEC_CNT_THRESH_EV . . . . .	807
6.30.2.53 VTSS_PHY_10G_TX_CHAR_ENC_CNT_THRESH_EV . . . . .	807
6.30.2.54 VTSS_PHY_10G_RX_BLK_DEC_CNT_THRESH_EV . . . . .	807
6.30.2.55 VTSS_PHY_10G_TX_BLK_ENC_CNT_THRESH_EV . . . . .	807
6.30.2.56 VTSS_PHY_10G_RX_SEQ_CNT_THRESH_EV . . . . .	808
6.30.2.57 VTSS_PHY_10G_TX_SEQ_CNT_THRESH_EV . . . . .	808
6.30.2.58 VTSS_PHY_10G_FEC_UNFIXED_CNT_THRESH_EV . . . . .	808
6.30.2.59 VTSS_PHY_10G_FEC_FIXED_CNT_THRESH_EV . . . . .	808
6.30.2.60 VTSS_PHY_10G_HIGHBER_EV . . . . .	808
6.30.2.61 VTSS_PHY_10G_RX_LINK_STAT_EV [1/2] . . . . .	809
6.30.2.62 VTSS_PHY_10G_RX_LINK_STAT_EV [2/2] . . . . .	809
6.30.2.63 VTSS_PHY_10G_GPIO_INT_AGG0_EV . . . . .	809
6.30.2.64 VTSS_PHY_10G_GPIO_INT_AGG1_EV . . . . .	809
6.30.2.65 VTSS_PHY_10G_GPIO_INT_AGG2_EV . . . . .	809
6.30.2.66 VTSS_PHY_10G_GPIO_INT_AGG3_EV . . . . .	810
6.30.2.67 VTSS_PHY_1G_LINE_AUTONEG_RESTART_EV . . . . .	810
6.30.2.68 VTSS_PHY_1G_HOST_AUTONEG_RESTART_EV . . . . .	810
6.30.2.69 VTSS_PHY_10G_LINE_MAC_LOCAL_FAULT_EV . . . . .	810
6.30.2.70 VTSS_PHY_10G_HOST_MAC_LOCAL_FAULT_EV . . . . .	810
6.30.2.71 VTSS_PHY_10G_LINE_MAC_REMOTE_FAULT_EV . . . . .	811
6.30.2.72 VTSS_PHY_10G_HOST_MAC_REMOTE_FAULT_EV . . . . .	811
6.30.3 Typedef Documentation . . . . .	811
6.30.3.1 vtss_gpio_10g_no_t . . . . .	811
6.30.3.2 ckout_sel_t . . . . .	811
6.30.3.3 vtss_32_cntr_t . . . . .	811
6.30.3.4 vtss_phy_10g_event_t . . . . .	812
6.30.3.5 vtss_phy_10g_extnd_event_t . . . . .	812
6.30.4 Enumeration Type Documentation . . . . .	812
6.30.4.1 oper_mode_t . . . . .	812
6.30.4.2 vtss_wrefclk_t . . . . .	813

6.30.4.3 vtss_phy_interface_mode . . . . .	813
6.30.4.4 vtss_recvrd_t . . . . .	814
6.30.4.5 vtss_recvrdclk_cdr_div_t . . . . .	814
6.30.4.6 vtss_srefclk_div_t . . . . .	814
6.30.4.7 vtss_wref_clk_div_t . . . . .	815
6.30.4.8 apc_ib_regulator_t . . . . .	815
6.30.4.9 ddr_mode_t . . . . .	815
6.30.4.10 clk_mstr_t . . . . .	816
6.30.4.11 vtss_rptr_rate_t . . . . .	816
6.30.4.12 vtss_phy_10g_media_t . . . . .	817
6.30.4.13 vtss_phy_6g_link_partner_distance_t . . . . .	817
6.30.4.14 vtss_phy_10g_ib_apc_op_mode_t . . . . .	817
6.30.4.15 vtss_channel_t . . . . .	818
6.30.4.16 vtss_recvrd_clkout_t . . . . .	818
6.30.4.17 vtss_phy_10g_srefclk_freq_t . . . . .	819
6.30.4.18 vtss_phy_10g_ckout_freq_t . . . . .	819
6.30.4.19 vtss_ckout_data_sel_t . . . . .	819
6.30.4.20 vtss_phy_10g_squelch_src_t . . . . .	820
6.30.4.21 vtss_phy_10g_clk_sel_t . . . . .	821
6.30.4.22 vtss_phy_10g_recvrd_clk_sel_t . . . . .	822
6.30.4.23 ckout_sel_ . . . . .	822
6.30.4.24 vtss_phy_10g_sckout_freq_t . . . . .	822
6.30.4.25 vtss_phy_10g_rx_macro_t . . . . .	823
6.30.4.26 vtss_phy_10g_tx_macro_t . . . . .	823
6.30.4.27 vtss_phy_10g_clause_37_remote_fault_t . . . . .	823
6.30.4.28 vtss_lb_type_t . . . . .	824
6.30.4.29 vtss_phy_10g_power_t . . . . .	825
6.30.4.30 vtss_phy_10g_failover_mode_t . . . . .	825
6.30.4.31 vtss_phy_10g_auto_failover_event_t . . . . .	825
6.30.4.32 vtss_phy_10g_auto_failover_filter_t . . . . .	826

6.30.4.33 vtss_phy_10g_vscope_scan_t . . . . .	826
6.30.4.34 vtss_phy_10g_pkt_mon_rst_t . . . . .	827
6.30.4.35 vtss_10g_phy_gpio_t . . . . .	827
6.30.4.36 vtss_gpio_10g_gpio_intr_sgnl_t . . . . .	828
6.30.4.37 vtss_gpio_10g_chan_intrpt_t . . . . .	830
6.30.4.38 vtss_gpio_10g_aggr_intrpt_t . . . . .	831
6.30.4.39 vtss_gpio_10g_input_t . . . . .	831
6.30.5 Function Documentation . . . . .	832
6.30.5.1 vtss_phy_10g_mode_get() . . . . .	832
6.30.5.2 vtss_phy_10g_init() . . . . .	832
6.30.5.3 vtss_phy_10g_mode_set() . . . . .	834
6.30.5.4 vtss_phy_10g_ib_conf_set() . . . . .	834
6.30.5.5 vtss_phy_10g_ib_conf_get() . . . . .	835
6.30.5.6 vtss_phy_10g_ib_status_get() . . . . .	835
6.30.5.7 vtss_phy_10g_apc_conf_set() . . . . .	836
6.30.5.8 vtss_phy_10g_apc_conf_get() . . . . .	836
6.30.5.9 vtss_phy_10g_apc_status_get() . . . . .	837
6.30.5.10 vtss_phy_10g_apc_restart() . . . . .	837
6.30.5.11 vtss_phy_10g_jitter_conf_set() . . . . .	838
6.30.5.12 vtss_phy_10g_jitter_conf_get() . . . . .	838
6.30.5.13 vtss_phy_10g_jitter_status_get() . . . . .	839
6.30.5.14 vtss_phy_10g_syncce_clkout_get() . . . . .	839
6.30.5.15 vtss_phy_10g_syncce_clkout_set() . . . . .	840
6.30.5.16 vtss_phy_10g_xfp_clkout_get() . . . . .	840
6.30.5.17 vtss_phy_10g_xfp_clkout_set() . . . . .	841
6.30.5.18 vtss_phy_10g_rxckout_get() . . . . .	841
6.30.5.19 vtss_phy_10g_rxckout_set() . . . . .	842
6.30.5.20 vtss_phy_10g_txckout_get() . . . . .	842
6.30.5.21 vtss_phy_10g_txckout_set() . . . . .	842
6.30.5.22 vtss_phy_10g_srefclk_conf_get() . . . . .	843

6.30.5.23 vtss_phy_10g_srefclk_conf_set()	843
6.30.5.24 vtss_phy_10g_sckout_conf_set()	844
6.30.5.25 vtss_phy_10g_ckout_conf_set()	844
6.30.5.26 vtss_phy_10g_line_clk_conf_set()	845
6.30.5.27 vtss_phy_10g_host_clk_conf_set()	845
6.30.5.28 vtss_phy_10g_line_recvrd_clk_conf_set()	846
6.30.5.29 vtss_phy_10g_host_recvrd_clk_conf_set()	846
6.30.5.30 vtss_phy_10g_lane_sync_set()	847
6.30.5.31 vtss_phy_10g_debug_register_dump()	847
6.30.5.32 vtss_phy_10g_base_kr_train_aneg_get()	848
6.30.5.33 vtss_phy_10g_base_kr_train_aneg_set()	848
6.30.5.34 vtss_phy_10g_base_host_kr_train_aneg_set()	848
6.30.5.35 vtss_phy_10g_base_kr_conf_get()	849
6.30.5.36 vtss_phy_10g_base_kr_conf_set()	849
6.30.5.37 vtss_phy_10g_base_kr_host_conf_get()	850
6.30.5.38 vtss_phy_10g_base_kr_host_conf_set()	850
6.30.5.39 vtss_phy_10g_kr_status_get()	851
6.30.5.40 vtss_phy_10g_ob_status_get()	851
6.30.5.41 vtss_phy_10g_status_get()	852
6.30.5.42 vtss_phy_10g_serdes_status_get()	852
6.30.5.43 vtss_phy_10g_reset()	853
6.30.5.44 vtss_phy_10g_clause_37_status_get()	853
6.30.5.45 vtss_phy_10g_clause_37_control_get()	853
6.30.5.46 vtss_phy_10g_clause_37_control_set()	854
6.30.5.47 vtss_phy_10g_loopback_set()	854
6.30.5.48 vtss_phy_10g_loopback_get()	855
6.30.5.49 vtss_phy_10g_cnt_get()	855
6.30.5.50 vtss_phy_10g_power_get()	856
6.30.5.51 vtss_phy_10g_power_set()	856
6.30.5.52 vtss_phy_10G_is_valid()	857

6.30.5.53 vtss_phy_10g_failover_set()	857
6.30.5.54 vtss_phy_10g_failover_get()	857
6.30.5.55 vtss_phy_10g_auto_failover_set()	858
6.30.5.56 vtss_phy_10g_auto_failover_get()	858
6.30.5.57 vtss_phy_10g_vscope_conf_set()	859
6.30.5.58 vtss_phy_10g_vscope_conf_get()	859
6.30.5.59 vtss_phy_10g_vscope_scan_status_get()	860
6.30.5.60 vtss_phy_10g_pcs_prbs_gen_conf_set()	860
6.30.5.61 vtss_phy_10g_pcs_prbs_gen_conf_get()	861
6.30.5.62 vtss_phy_10g_pcs_prbs_mon_conf_set()	861
6.30.5.63 vtss_phy_10g_pcs_prbs_mon_conf_get()	862
6.30.5.64 vtss_phy_10g_pcs_prbs_mon_status_get()	862
6.30.5.65 vtss_phy_10g_prbs_gen_conf()	863
6.30.5.66 vtss_phy_10g_prbs_gen_conf_get()	863
6.30.5.67 vtss_phy_10g_prbs_mon_conf()	863
6.30.5.68 vtss_phy_10g_prbs_mon_conf_get()	864
6.30.5.69 vtss_phy_10g_prbs_mon_status_get()	864
6.30.5.70 vtss_phy_10g_pkt_gen_conf()	865
6.30.5.71 vtss_phy_10g_pkt_mon_conf()	865
6.30.5.72 vtss_phy_10g_pkt_mon_counters_get()	866
6.30.5.73 vtss_phy_10g_id_get()	866
6.30.5.74 vtss_phy_10g_gpio_mode_set()	867
6.30.5.75 vtss_phy_10g_gpio_mode_get()	867
6.30.5.76 vtss_phy_10g_gpio_read()	868
6.30.5.77 vtss_phy_10g_gpio_write()	868
6.30.5.78 vtss_phy_10g_event_enable_set()	869
6.30.5.79 vtss_phy_10g_event_enable_get()	869
6.30.5.80 vtss_phy_10g_extended_event_enable_get()	869
6.30.5.81 vtss_phy_10g_event_poll()	870
6.30.5.82 vtss_phy_10g_pcs_status_get()	870

6.30.5.83 vtss_phy_10g_extended_event_poll()	871
6.30.5.84 vtss_phy_10g_extended_event_enable_set()	871
6.30.5.85 vtss_phy_10g_poll_1sec()	872
6.30.5.86 vtss_phy_10g_edc_fw_status_get()	872
6.30.5.87 vtss_phy_10g_fc_buffer_reset()	872
6.30.5.88 vtss_phy_10g_csr_read()	873
6.30.5.89 vtss_phy_10g_csr_write()	873
6.30.5.90 vtss_phy_warm_start_10g_failed_get()	874
6.30.5.91 vtss_phy_10g_sgmii_mode_set()	874
6.30.5.92 vtss_phy_10g_i2c_read()	875
6.30.5.93 vtss_phy_10g_i2c_write()	875
6.30.5.94 vtss_phy_10g_get_user_data()	875
6.31 vtss_api/include/vtss_phy_api.h File Reference	876
6.31.1 Detailed Description	886
6.31.2 Macro Definition Documentation	886
6.31.2.1 MAX_CFG_BUF_SIZE	886
6.31.2.2 MAX_STAT_BUF_SIZE	886
6.31.2.3 VTSS_PHY_POWER_ACTIPHY_BIT	886
6.31.2.4 VTSS_PHY_POWER_DYNAMIC_BIT	887
6.31.2.5 VTSS_PHY_ACTIPHY_PWR	887
6.31.2.6 VTSS_PHY_LINK_DOWN_PWR	887
6.31.2.7 VTSS_PHY_LINK_UP_FULL_PWR	887
6.31.2.8 VTSS_PHY_RECov_CLK1	887
6.31.2.9 VTSS_PHY_RECov_CLK2	888
6.31.2.10 VTSS_PHY_RECov_CLK_NUM	888
6.31.2.11 VTSS_PHY_PAGE_STANDARD	888
6.31.2.12 VTSS_PHY_PAGE_EXTENDED	888
6.31.2.13 VTSS_PHY_PAGE_EXTENDED_2	888
6.31.2.14 VTSS_PHY_PAGE_EXTENDED_3	889
6.31.2.15 VTSS_PHY_PAGE_EXTENDED_4	889

6.31.2.16 VTSS_PHY_PAGE_GPIO . . . . .	889
6.31.2.17 VTSS_PHY_PAGE_1588 . . . . .	889
6.31.2.18 VTSS_PHY_PAGE_MACSEC . . . . .	889
6.31.2.19 VTSS_PHY_PAGE_TEST . . . . .	890
6.31.2.20 VTSS_PHY_PAGE_TR . . . . .	890
6.31.2.21 VTSS_PHY_PAGE_0x2DAF . . . . .	890
6.31.2.22 VTSS_PHY_REG_STANDARD . . . . .	890
6.31.2.23 VTSS_PHY_REG_EXTENDED . . . . .	890
6.31.2.24 VTSS_PHY_REG_GPIO . . . . .	891
6.31.2.25 VTSS_PHY_REG_TEST . . . . .	891
6.31.2.26 VTSS_PHY_REG_TR . . . . .	891
6.31.2.27 VTSS_PHY_LINK_LOS_EV . . . . .	891
6.31.2.28 VTSS_PHY_LINK_FFAIL_EV . . . . .	891
6.31.2.29 VTSS_PHY_LINK_AMS_EV . . . . .	892
6.31.2.30 VTSS_PHY_LINK_SPEED_STATE_CHANGE_EV . . . . .	892
6.31.2.31 VTSS_PHY_LINK_FDX_STATE_CHANGE_EV . . . . .	892
6.31.2.32 VTSS_PHY_LINK_AUTO_NEG_ERROR_EV . . . . .	892
6.31.2.33 VTSS_PHY_LINK_AUTO_NEG_COMPLETE_EV . . . . .	892
6.31.2.34 VTSS_PHY_LINK_INLINE_POW_DEV_DETECT_EV . . . . .	893
6.31.2.35 VTSS_PHY_LINK_SYMBOL_ERR_INT_EV . . . . .	893
6.31.2.36 VTSS_PHY_LINK_TX_FIFO_OVERFLOW_INT_EV . . . . .	893
6.31.2.37 VTSS_PHY_LINK_RX_FIFO_OVERFLOW_INT_EV . . . . .	893
6.31.2.38 VTSS_PHY_LINK_FALSE_CARRIER_INT_EV . . . . .	893
6.31.2.39 VTSS_PHY_LINK_LINK_SPEED_DS_DETECT_EV . . . . .	894
6.31.2.40 VTSS_PHY_LINK_MASTER_SLAVE_RES_ERR_EV . . . . .	894
6.31.2.41 VTSS_PHY_LINK_RX_ER_INT_EV . . . . .	894
6.31.2.42 VTSS_PHY_LINK_EXTENDED_REG_INT_EV . . . . .	894
6.31.2.43 VTSS_PHY_LINK_WAKE_ON_LAN_INT_EV . . . . .	894
6.31.2.44 VTSS_PHY_LINK_EXT_EEE_WAKE_ERR_EV . . . . .	895
6.31.2.45 VTSS_PHY_LINK_EXT_EEE_WAIT_TS_EV . . . . .	895

6.31.2.46 VTSS_PHY_LINK_EXT_EEE_WAIT_RX_TQ_EV . . . . .	895
6.31.2.47 VTSS_PHY_LINK_EXT_EEE_LINKFAIL_EV . . . . .	895
6.31.2.48 VTSS_PHY_LINK_EXT_RR_SW_COMPL_EV . . . . .	895
6.31.2.49 VTSS_PHY_LINK_EXT_MACSEC_HOST_MAC_EV . . . . .	896
6.31.2.50 VTSS_PHY_LINK_EXT_MACSEC_LINE_MAC_EV . . . . .	896
6.31.2.51 VTSS_PHY_LINK_EXT_MACSEC_FC_BUFF_EV . . . . .	896
6.31.2.52 VTSS_PHY_LINK_EXT_MACSEC_INGRESS_EV . . . . .	896
6.31.2.53 VTSS_PHY_LINK_EXT_MACSEC_EGRESS_EV . . . . .	896
6.31.2.54 VTSS_PHY_LINK_EXT_MEM_INT_RING_EV . . . . .	897
6.31.2.55 MAX_WOL_MAC_ADDR_SIZE . . . . .	897
6.31.2.56 MAX_WOL_PASSWD_SIZE . . . . .	897
6.31.2.57 MIN_WOL_PASSWD_SIZE . . . . .	897
6.31.3 Typedef Documentation . . . . .	897
6.31.3.1 vtss_phy_recov_clk_t . . . . .	897
6.31.3.2 vtss_phy_led_intensity . . . . .	898
6.31.4 Enumeration Type Documentation . . . . .	898
6.31.4.1 vtss_phy_led_mode_t . . . . .	898
6.31.4.2 vtss_phy_media_interface_t . . . . .	899
6.31.4.3 vtss_phy_mdi_t . . . . .	899
6.31.4.4 rgmii_skew_delay_psec_t . . . . .	900
6.31.4.5 vtss_phy_forced_reset_t . . . . .	900
6.31.4.6 vtss_phy_pkt_mode_t . . . . .	901
6.31.4.7 vtss_phy_mode_t . . . . .	901
6.31.4.8 vtss_phy_fast_link_fail_t . . . . .	901
6.31.4.9 vtss_phy_sigdet_polarity_t . . . . .	902
6.31.4.10 vtss_phy_unidirectional_t . . . . .	902
6.31.4.11 vtss_phy_mac_serdes_pcs_sgmii_pre . . . . .	902
6.31.4.12 vtss_phy_media_rem_fault_t . . . . .	903
6.31.4.13 vtss_phy_media_force_ams_sel_t . . . . .	903
6.31.4.14 vtss_phy_clk_source_t . . . . .	903

6.31.4.15 <code>vtss_phy_freq_t</code>	904
6.31.4.16 <code>vtss_phy_clk_squelch</code>	904
6.31.4.17 <code>vtss_phy_gpio_mode_t</code>	904
6.31.4.18 <code>lb_type</code>	905
6.31.4.19 <code>vtss_wol_passwd_len_type_t</code>	905
6.31.4.20 <code>vtss_fefi_mode_t</code>	906
6.31.5 Function Documentation	906
6.31.5.1 <code>vtss_phy_pre_reset()</code>	906
6.31.5.2 <code>vtss_phy_post_reset()</code>	906
6.31.5.3 <code>vtss_phy_pre_system_reset()</code>	907
6.31.5.4 <code>vtss_phy_reset()</code>	907
6.31.5.5 <code>vtss_phy_reset_get()</code>	908
6.31.5.6 <code>vtss_phy_chip_temp_get()</code>	908
6.31.5.7 <code>vtss_phy_chip_temp_init()</code>	908
6.31.5.8 <code>vtss_phy_conf_get()</code>	910
6.31.5.9 <code>vtss_phy_conf_set()</code>	910
6.31.5.10 <code>vtss_phy_status_get()</code>	911
6.31.5.11 <code>vtss_phy_cl37_lp_abil_get()</code>	911
6.31.5.12 <code>vtss_phy_id_get()</code>	912
6.31.5.13 <code>vtss_phy_conf_1g_get()</code>	912
6.31.5.14 <code>vtss_phy_conf_1g_set()</code>	912
6.31.5.15 <code>vtss_phy_status_1g_get()</code>	913
6.31.5.16 <code>vtss_phy_power_conf_get()</code>	913
6.31.5.17 <code>vtss_phy_power_conf_set()</code>	914
6.31.5.18 <code>vtss_phy_power_status_get()</code>	914
6.31.5.19 <code>vtss_phy_clock_conf_set()</code>	915
6.31.5.20 <code>vtss_phy_clock_conf_get()</code>	915
6.31.5.21 <code>vtss_phy_i2c_read()</code>	916
6.31.5.22 <code>vtss_phy_i2c_write()</code>	916
6.31.5.23 <code>vtss_phy_read()</code>	917

6.31.5.24 vtss_phy_read_page()	917
6.31.5.25 vtss_phy_mmd_read()	918
6.31.5.26 vtss_phy_mmd_write()	918
6.31.5.27 vtss_phy_write()	919
6.31.5.28 vtss_phy_write_masked()	919
6.31.5.29 vtss_phy_write_masked_page()	920
6.31.5.30 vtss_phy_gpio_mode()	920
6.31.5.31 vtss_phy_gpio_get()	921
6.31.5.32 vtss_phy_gpio_set()	921
6.31.5.33 vtss_phy_veriphy_start()	922
6.31.5.34 vtss_phy_veriphy_get()	922
6.31.5.35 vtss_phy_led_mode_set()	923
6.31.5.36 vtss_phy_led_intensity_set()	923
6.31.5.37 vtss_phy_led_intensity_get()	923
6.31.5.38 vtss_phy_enhanced_led_control_init()	924
6.31.5.39 vtss_phy_enhanced_led_control_init_get()	924
6.31.5.40 vtss_phy_coma_mode_disable()	925
6.31.5.41 vtss_phy_coma_mode_enable()	925
6.31.5.42 vga_adc_debug()	925
6.31.5.43 vtss_phy_port_eee_capable()	926
6.31.5.44 vtss_phy_eee_ena()	926
6.31.5.45 vtss_phy_eee_conf_get()	927
6.31.5.46 vtss_phy_eee_conf_set()	927
6.31.5.47 vtss_phy_eee_power_save_state_get()	928
6.31.5.48 vtss_phy_eee_link_partner_advertisements_get()	928
6.31.5.49 vtss_phy_event_enable_set()	928
6.31.5.50 vtss_phy_event_enable_get()	929
6.31.5.51 vtss_phy_event_poll()	929
6.31.5.52 vtss_squelch_workaround()	930
6.31.5.53 vtss_phy_csr_wr()	930

6.31.5.54 vtss_phy_csr_rd()	931
6.31.5.55 vtss_phy_statistic_get()	931
6.31.5.56 vtss_phy_do_page_chk_set()	932
6.31.5.57 vtss_phy_do_page_chk_get()	932
6.31.5.58 vtss_phy_loopback_set()	933
6.31.5.59 vtss_phy_loopback_get()	933
6.31.5.60 vtss_phy_is_8051_crc_ok()	934
6.31.5.61 vtss_phy_cfg_ob_post0()	934
6.31.5.62 vtss_phy_cfg_ib_cterm()	934
6.31.5.63 vtss_phy_atom12_patch_settings_get()	935
6.31.5.64 vtss_phy_reg_decode_status()	935
6.31.5.65 vtss_phy_flowcontrol_decode_status()	936
6.31.5.66 vtss_phy_debug_stat_print()	936
6.31.5.67 vtss_phy_warm_start_failed_get()	937
6.31.5.68 vtss_phy_debug_phyinfo_print()	937
6.31.5.69 vtss_phy_debug_register_dump()	938
6.31.5.70 vtss_phy_detect_base_ports()	938
6.31.5.71 vtss_phy_ext_connector_loopback()	939
6.31.5.72 vtss_phy_serdes_sgmii_loopback()	939
6.31.5.73 vtss_phy_serdes_fmedia_loopback()	940
6.31.5.74 vtss_phy_debug_regdump_print()	940
6.31.5.75 vtss_phy_wol_enable()	941
6.31.5.76 vtss_phy_wol_conf_get()	941
6.31.5.77 vtss_phy_wol_conf_set()	941
6.31.5.78 vtss_phy_patch_settings_get()	942
6.31.5.79 vtss_phy_serdes6g_rcpll_status_get()	942
6.31.5.80 vtss_phy_serdes1g_rcpll_status_get()	943
6.31.5.81 vtss_phy_lcpll_status_get()	943
6.31.5.82 vtss_phy_reset_lcpll()	944
6.31.5.83 vtss_phy_sd6g_ob_post_rd()	944

6.31.5.84 vtss_phy_sd6g_ob_post_wr() . . . . .	945
6.31.5.85 vtss_phy_sd6g_ob_lev_rd() . . . . .	945
6.31.5.86 vtss_phy_sd6g_ob_lev_wr() . . . . .	946
6.31.5.87 vtss_phy_mac_media_inhibit_odd_start() . . . . .	946
6.31.5.88 vtss_phy_fefi_get() . . . . .	947
6.31.5.89 vtss_phy_fefi_set() . . . . .	947
6.31.5.90 vtss_phy_fefi_detect() . . . . .	947
6.31.5.91 vtss_phy_mse_100m_get() . . . . .	948
6.31.5.92 vtss_phy_mse_1000m_get() . . . . .	948
6.31.5.93 vtss_phy_read_tr_addr() . . . . .	949
6.31.5.94 vtss_phy_is_viper_revB() . . . . .	949
6.31.5.95 vtss_phy_ext_event_poll() . . . . .	950
6.31.5.96 vtss_phy_status_inst_poll() . . . . .	950
6.31.5.97 vtss_phy_macsec_csr_sd6g_rd() . . . . .	951
6.31.5.98 vtss_phy_macsec_csr_sd6g_wr() . . . . .	951
6.32 vtss_api/include/vtss_phy_ts_api.h File Reference . . . . .	952
6.32.1 Detailed Description . . . . .	961
6.32.2 Macro Definition Documentation . . . . .	961
6.32.2.1 VTSS_PHY_TS_FIFO_SIG_SRC_IP . . . . .	961
6.32.2.2 VTSS_PHY_TS_FIFO_SIG_DEST_IP . . . . .	961
6.32.2.3 VTSS_PHY_TS_FIFO_SIG_MSG_TYPE . . . . .	961
6.32.2.4 VTSS_PHY_TS_FIFO_SIG_DOMAIN_NUM . . . . .	962
6.32.2.5 VTSS_PHY_TS_FIFO_SIG_SOURCE_PORT_ID . . . . .	962
6.32.2.6 VTSS_PHY_TS_FIFO_SIG_SEQ_ID . . . . .	962
6.32.2.7 VTSS_PHY_TS_FIFO_SIG_DEST_MAC . . . . .	962
6.32.2.8 VTSS_PHY_TS_SIG_LEN . . . . .	962
6.32.2.9 VTSS_PHY_TS_SIG_TIME_STAMP_LEN . . . . .	963
6.32.2.10 VTSS_PHY_TS_SIG_DOMAIN_NUM_LEN . . . . .	963
6.32.2.11 VTSS_PHY_TS_SIG_SEQ_ID_LEN . . . . .	963
6.32.2.12 VTSS_PHY_TS_SIG_MSG_TYPE_LEN . . . . .	963

6.32.2.13 VTSS_PHY_TS_SIG_SOURCE_PORT_ID_LEN . . . . .	963
6.32.2.14 VTSS_PHY_TS_SIG_SEQUENCE_ID_LEN . . . . .	964
6.32.2.15 VTSS_PHY_TS_SIG_DEST_IP_LEN . . . . .	964
6.32.2.16 VTSS_PHY_TS_SIG_SRC_IP_LEN . . . . .	964
6.32.2.17 VTSS_PHY_TS_SIG_DEST_MAC_LEN . . . . .	964
6.32.2.18 VTSS_PTP_IP_1588_VERSION_2_1 . . . . .	964
6.32.2.19 VTSS_PHY_TS_ETH_ADDR_MATCH_48BIT . . . . .	965
6.32.2.20 VTSS_PHY_TS_ETH_ADDR_MATCH_ANY_UNICAST . . . . .	965
6.32.2.21 VTSS_PHY_TS_ETH_ADDR_MATCH_ANY_MULTICAST . . . . .	965
6.32.2.22 VTSS_PHY_TS_ETH_MATCH_DEST_ADDR . . . . .	965
6.32.2.23 VTSS_PHY_TS_ETH_MATCH_SRC_ADDR . . . . .	965
6.32.2.24 VTSS_PHY_TS_ETH_MATCH_SRC_OR_DEST . . . . .	966
6.32.2.25 VTSS_PHY_TS_TAG_TYPE_C . . . . .	966
6.32.2.26 VTSS_PHY_TS_TAG_TYPE_S . . . . .	966
6.32.2.27 VTSS_PHY_TS_TAG_TYPE_I . . . . .	966
6.32.2.28 VTSS_PHY_TS_TAG_TYPE_B . . . . .	966
6.32.2.29 VTSS_PHY_TS_TAG_RANGE_NONE . . . . .	967
6.32.2.30 VTSS_PHY_TS_TAG_RANGE_OUTER . . . . .	967
6.32.2.31 VTSS_PHY_TS_TAG_RANGE_INNER . . . . .	967
6.32.2.32 VTSS_PHY_TS_IP_VER_4 . . . . .	967
6.32.2.33 VTSS_PHY_TS_IP_VER_6 . . . . .	967
6.32.2.34 VTSS_PHY_TS_IP_MATCH_SRC . . . . .	968
6.32.2.35 VTSS_PHY_TS_IP_MATCH_DEST . . . . .	968
6.32.2.36 VTSS_PHY_TS_IP_MATCH_SRC_OR_DEST . . . . .	968
6.32.2.37 VTSS_PHY_TS_MPLS_STACK_DEPTH_1 . . . . .	968
6.32.2.38 VTSS_PHY_TS_MPLS_STACK_DEPTH_2 . . . . .	968
6.32.2.39 VTSS_PHY_TS_MPLS_STACK_DEPTH_3 . . . . .	969
6.32.2.40 VTSS_PHY_TS_MPLS_STACK_DEPTH_4 . . . . .	969
6.32.2.41 VTSS_PHY_TS_MPLS_STACK_REF_POINT_TOP . . . . .	969
6.32.2.42 VTSS_PHY_TS_MPLS_STACK_REF_POINT_END . . . . .	969

6.32.2.43 VTSS_PHY_TS_ENG_FLOW_VALID_FOR_CH0 . . . . .	969
6.32.2.44 VTSS_PHY_TS_ENG_FLOW_VALID_FOR_CH1 . . . . .	970
6.32.2.45 VTSS_PHY_TS_INGR_ENGINE_ERR . . . . .	970
6.32.2.46 VTSS_PHY_TS_INGR_RW_PREAM_ERR . . . . .	970
6.32.2.47 VTSS_PHY_TS_INGR_RW_FCS_ERR . . . . .	970
6.32.2.48 VTSS_PHY_TS_EGR_ENGINE_ERR . . . . .	970
6.32.2.49 VTSS_PHY_TS_EGR_RW_FCS_ERR . . . . .	971
6.32.2.50 VTSS_PHY_TS_EGR_TIMESTAMP_CAPTURED . . . . .	971
6.32.2.51 VTSS_PHY_TS_EGR_FIFO_OVERFLOW . . . . .	971
6.32.2.52 VTSS_PHY_TS_DATA_IN_RSRVD_FIELD . . . . .	971
6.32.2.53 VTSS_PHY_TS_LTC_NEW_PPS_INTRPT . . . . .	971
6.32.2.54 VTSS_PHY_TS_LTC_LOAD_SAVE_NEW_TOD . . . . .	972
6.32.2.55 VTSS_PHY_TS_8487_XAUI_SEL_0 . . . . .	972
6.32.2.56 VTSS_PHY_TS_8487_XAUI_SEL_1 . . . . .	972
6.32.2.57 VTSS_PHY_TS_INGR_DATAPATH_RESET . . . . .	972
6.32.2.58 VTSS_PHY_TS_EGR_DATAPATH_RESET . . . . .	972
6.32.2.59 VTSS_PHY_TS_INGR_LTC1_RESET . . . . .	973
6.32.2.60 VTSS_PHY_TS_EGR_LTC2_RESET . . . . .	973
6.32.2.61 VTSS_PHY_TS_EGR_FIFO_RESET . . . . .	973
6.32.3 Typedef Documentation . . . . .	973
6.32.3.1 vtss_phy_ts_alt_clock_mode_t . . . . .	973
6.32.3.2 vtss_phy_ts_scaled_ppb_t . . . . .	973
6.32.3.3 vtss_phy_ts_fifo_sig_mask_t . . . . .	974
6.32.3.4 vtss_phy_ts_fifo_read . . . . .	974
6.32.3.5 vtss_phy_ts_engine_channel_map_t . . . . .	974
6.32.3.6 vtss_phy_ts_event_t . . . . .	974
6.32.3.7 vtss_phy_ts_8487_xaui_sel_t . . . . .	974
6.32.3.8 vtss_phy_ts_soft_reset_t . . . . .	975
6.32.4 Enumeration Type Documentation . . . . .	975
6.32.4.1 vtss_phy_ts_todadj_status_t . . . . .	975

6.32.4.2 vtss_phy_ts_fifo_status_t . . . . .	975
6.32.4.3 vtss_phy_ts_encap_t . . . . .	976
6.32.4.4 vtss_phy_ts_engine_t . . . . .	976
6.32.4.5 vtss_phy_ts_engine_flow_match_t . . . . .	976
6.32.4.6 vtss_phy_ts_ptp_clock_mode_t . . . . .	977
6.32.4.7 vtss_phy_ts_ptp_delaym_type_t . . . . .	977
6.32.4.8 vtss_phy_ts_y1731_oam_delaym_type_t . . . . .	978
6.32.4.9 vtss_phy_ts_ietf_mpls_ach_oam_delaym_type_t . . . . .	978
6.32.4.10 vtss_phy_ts_ietf_mpls_ach_oam_ts_format_t . . . . .	979
6.32.4.11 vtss_phy_ts_action_format . . . . .	979
6.32.4.12 vtss_phy_ts_clockfreq_t . . . . .	979
6.32.4.13 vtss_phy_ts_clock_src_t . . . . .	980
6.32.4.14 vtss_phy_ts_rxtimestamp_pos_t . . . . .	980
6.32.4.15 vtss_phy_ts_rxtimestamp_len_t . . . . .	981
6.32.4.16 vtss_phy_ts_fifo_mode_t . . . . .	981
6.32.4.17 vtss_phy_ts_fifo_timestamp_len_t . . . . .	981
6.32.4.18 vtss_phy_ts_tc_op_mode_t . . . . .	982
6.32.4.19 vtss_phy_ts_nphase_sampler_t . . . . .	982
6.32.4.20 vtss_phy_ts_ptp_message_type_t . . . . .	983
6.32.5 Function Documentation . . . . .	983
6.32.5.1 vtss_phy_ts_alt_clock_saved_get() . . . . .	983
6.32.5.2 vtss_phy_ts_alt_clock_mode_get() . . . . .	983
6.32.5.3 vtss_phy_ts_alt_clock_mode_set() . . . . .	984
6.32.5.4 vtss_phy_ts_pps_conf_set() . . . . .	984
6.32.5.5 vtss_phy_ts_pps_conf_get() . . . . .	985
6.32.5.6 vtss_phy_ts_ingress_latency_set() . . . . .	985
6.32.5.7 vtss_phy_ts_ingress_latency_get() . . . . .	986
6.32.5.8 vtss_phy_ts_egress_latency_set() . . . . .	986
6.32.5.9 vtss_phy_ts_egress_latency_get() . . . . .	987
6.32.5.10 vtss_phy_ts_path_delay_set() . . . . .	987

6.32.5.11 vtss_phy_ts_path_delay_get()	988
6.32.5.12 vtss_phy_ts_delay_asymmetry_set()	988
6.32.5.13 vtss_phy_ts_delay_asymmetry_get()	989
6.32.5.14 vtss_phy_ts_ptptime_set()	989
6.32.5.15 vtss_phy_ts_ptptime_set_done()	990
6.32.5.16 vtss_phy_ts_ptptime_arm()	990
6.32.5.17 vtss_phy_ts_ptptime_get()	991
6.32.5.18 vtss_phy_ts_load_ptptime_get()	991
6.32.5.19 vtss_phy_ts_sertod_set()	991
6.32.5.20 vtss_phy_ts_sertod_get()	992
6.32.5.21 vtss_phy_ts_loadpulse_delay_set()	992
6.32.5.22 vtss_phy_ts_loadpulse_delay_get()	993
6.32.5.23 vtss_phy_ts_clock_rateadj_set()	993
6.32.5.24 vtss_phy_ts_clock_rateadj_get()	994
6.32.5.25 vtss_phy_ts_clock_rateadj_ppm_set()	994
6.32.5.26 vtss_phy_ts_clock_rateadj_ppm_get()	994
6.32.5.27 vtss_phy_ts_ptptime_adj1ns()	995
6.32.5.28 vtss_phy_ts_timeofday_offset_set()	995
6.32.5.29 vtss_phy_ts_ongoing_adjustment()	996
6.32.5.30 vtss_phy_ts_ltc_freq_synth_pulse_set()	996
6.32.5.31 vtss_phy_ts_ltc_freq_synth_pulse_get()	997
6.32.5.32 vtss_phy_daisy_conf_set()	997
6.32.5.33 vtss_phy_daisy_conf_get()	997
6.32.5.34 vtss_phy_ts_fifo_sig_set()	998
6.32.5.35 vtss_phy_ts_fifo_sig_get()	998
6.32.5.36 vtss_phy_ts_fifo_empty()	999
6.32.5.37 vtss_phy_ts_fifo_read_install()	999
6.32.5.38 vtss_phy_ts_fifo_read_cb_get()	1000
6.32.5.39 vtss_phy_ts_ingress_engine_init()	1000
6.32.5.40 vtss_phy_ts_ingress_engine_init_conf_get()	1001

6.32.5.41 vtss_phy_ts_ingress_engine_clear()	1001
6.32.5.42 vtss_phy_ts_egress_engine_init()	1002
6.32.5.43 vtss_phy_ts_egress_engine_init_conf_get()	1003
6.32.5.44 vtss_phy_ts_egress_engine_clear()	1003
6.32.5.45 vtss_phy_ts_ingress_engine_conf_set()	1004
6.32.5.46 vtss_phy_ts_ingress_engine_conf_get()	1004
6.32.5.47 vtss_phy_ts_egress_engine_conf_set()	1005
6.32.5.48 vtss_phy_ts_egress_engine_conf_get()	1005
6.32.5.49 vtss_phy_ts_ingress_engine_action_set()	1006
6.32.5.50 vtss_phy_ts_ingress_engine_action_get()	1006
6.32.5.51 vtss_phy_ts_egress_engine_action_set()	1008
6.32.5.52 vtss_phy_ts_egress_engine_action_get()	1008
6.32.5.53 vtss_phy_ts_event_enable_set()	1010
6.32.5.54 vtss_phy_ts_event_enable_get()	1010
6.32.5.55 vtss_phy_ts_event_poll()	1011
6.32.5.56 vtss_phy_ts_stats_get()	1011
6.32.5.57 vtss_phy_ts_correction_overflow_get()	1012
6.32.5.58 vtss_phy_ts_mode_set()	1012
6.32.5.59 vtss_phy_ts_mode_get()	1013
6.32.5.60 vtss_phy_ts_init()	1013
6.32.5.61 vtss_phy_ts_init_conf_get()	1013
6.32.5.62 vtss_phy_ts_nphase_status_get()	1014
6.32.5.63 vtss_phy_ts_hiacc_set()	1014
6.32.5.64 vtss_phy_ts_hiacc_get()	1015
6.32.5.65 vtss_phy_ts_block_soft_reset()	1015
6.32.5.66 vtss_phy_ts_phy_oper_mode_change()	1016
6.32.5.67 vtss_phy_1588_csr_reg_write()	1016
6.32.5.68 vtss_phy_1588_csr_reg_read()	1017
6.32.5.69 vtss_phy_ts_status_check()	1017
6.32.5.70 vtss_phy_ts_10g_extended_fifo_sync()	1018

6.32.5.71 <code>vtss_phy_ts_10g_fifo_sync()</code>	1018
6.32.5.72 <code>vtss_phy_ts_bypass_clear()</code>	1019
6.32.5.73 <code>vtss_phy_ts_viper_fifo_reset()</code>	1019
6.32.5.74 <code>vtss_phy_ts_tesla_tsp_fifo_sync()</code>	1019
6.32.5.75 <code>vtss_phy_1g_ts_fifo_sync()</code>	1020
6.32.5.76 <code>vtss_phy_1588_debug_reg_read()</code>	1021
6.32.5.77 <code>vtss_phy_ts_flow_clear_cf_set()</code>	1021
6.33 <code>vtss_api/include/vtss_port_api.h</code> File Reference	1022
6.33.1 Detailed Description	1024
6.33.2 Macro Definition Documentation	1024
6.33.2.1 <code>CHIP_PORT_UNUSED</code>	1024
6.33.2.2 <code>VTSS_FRAME_GAP_DEFAULT</code>	1025
6.33.2.3 <code>VTSS_MAX_FRAME_LENGTH_STANDARD</code>	1025
6.33.2.4 <code>VTSS_MAX_FRAME_LENGTH_MAX [1/2]</code>	1025
6.33.2.5 <code>VTSS_MAX_FRAME_LENGTH_MAX [2/2]</code>	1025
6.33.3 Enumeration Type Documentation	1025
6.33.3.1 <code>vtss_miim_controller_t</code>	1025
6.33.3.2 <code>vtss_internal_bw_t</code>	1026
6.33.3.3 <code>vtss_port_clause_37_remote_fault_t</code>	1026
6.33.3.4 <code>vtss_port_max_tags_t</code>	1027
6.33.3.5 <code>vtss_port_loop_t</code>	1027
6.33.3.6 <code>vtss_port_forward_t</code>	1027
6.33.4 Function Documentation	1028
6.33.4.1 <code>vtss_port_map_set()</code>	1028
6.33.4.2 <code>vtss_port_map_get()</code>	1028
6.33.4.3 <code>vtss_port_clause_37_control_get()</code>	1028
6.33.4.4 <code>vtss_port_clause_37_control_set()</code>	1029
6.33.4.5 <code>vtss_port_conf_set()</code>	1029
6.33.4.6 <code>vtss_port_conf_get()</code>	1030
6.33.4.7 <code>vtss_port_status_get()</code>	1030

6.33.4.8 vtss_port_counters_update()	1031
6.33.4.9 vtss_port_counters_clear()	1031
6.33.4.10 vtss_port_counters_get()	1031
6.33.4.11 vtss_port_basic_counters_get()	1032
6.33.4.12 vtss_port_forward_state_get()	1032
6.33.4.13 vtss_port_forward_state_set()	1033
6.33.4.14 vtss_port_ifh_conf_set()	1033
6.33.4.15 vtss_port_ifh_conf_get()	1034
6.33.4.16 vtss_miim_read()	1034
6.33.4.17 vtss_miim_write()	1035
6.33.4.18 vtss_port_mmd_read()	1035
6.33.4.19 vtss_port_mmd_read_inc()	1036
6.33.4.20 vtss_port_mmd_write()	1036
6.33.4.21 vtss_port_mmd_masked_write()	1037
6.33.4.22 vtss_mmd_read()	1037
6.33.4.23 vtss_mmd_write()	1038
6.34 vtss_api/include/vtss_qos_api.h File Reference	1038
6.34.1 Detailed Description	1040
6.34.2 Macro Definition Documentation	1040
6.34.2.1 VTSS_WRED_DPL_CNT	1041
6.34.2.2 VTSS_WRED_GROUP_CNT	1041
6.34.2.3 VTSS_PORT_POLICERS	1041
6.34.2.4 VTSS_PORT_POLICER_CPU_QUEUES	1041
6.34.2.5 VTSS_QCL_IDS	1041
6.34.2.6 VTSS_QCL_ID_START	1042
6.34.2.7 VTSS_QCL_ID_END	1042
6.34.2.8 VTSS_QCL_ARRAY_SIZE	1042
6.34.2.9 VTSS_QCE_ID_LAST	1042
6.34.3 Typedef Documentation	1042
6.34.3.1 vtss_red_v3_t	1042

6.34.4 Enumeration Type Documentation . . . . .	1042
6.34.4.1 vtss_wred_v2_max_t . . . . .	1042
6.34.4.2 vtss_tag_remark_mode_t . . . . .	1043
6.34.4.3 vtss_dscp_mode_t . . . . .	1043
6.34.4.4 vtss_dscp_emode_t . . . . .	1043
6.34.4.5 vtss_qce_type_t . . . . .	1044
6.34.5 Function Documentation . . . . .	1044
6.34.5.1 vtss_qos_conf_get() . . . . .	1044
6.34.5.2 vtss_qos_conf_set() . . . . .	1045
6.34.5.3 vtss_qos_port_conf_get() . . . . .	1045
6.34.5.4 vtss_qos_port_conf_set() . . . . .	1045
6.34.5.5 vtss_qce_init() . . . . .	1046
6.34.5.6 vtss_qce_add() . . . . .	1046
6.34.5.7 vtss_qce_del() . . . . .	1047
6.35 vtss_api/include/vtss_rab_api.h File Reference . . . . .	1047
6.35.1 Detailed Description . . . . .	1047
6.36 vtss_api/include/vtss_security_api.h File Reference . . . . .	1048
6.36.1 Detailed Description . . . . .	1050
6.36.2 Macro Definition Documentation . . . . .	1050
6.36.2.1 VTSS_ACE_ID_LAST . . . . .	1051
6.36.3 Enumeration Type Documentation . . . . .	1051
6.36.3.1 vtss_auth_state_t . . . . .	1051
6.36.3.2 vtss_acl_port_action_t . . . . .	1051
6.36.3.3 vtss_acl_ptp_action_t . . . . .	1051
6.36.3.4 vtss_acl_ptp_rsp_t . . . . .	1052
6.36.3.5 vtss_ace_type_t . . . . .	1052
6.36.3.6 vtss_ace_bit_t . . . . .	1053
6.36.4 Function Documentation . . . . .	1053
6.36.4.1 vtss_auth_port_state_get() . . . . .	1053
6.36.4.2 vtss_auth_port_state_set() . . . . .	1053

6.36.4.3 vtss_acl_policer_conf_get()	1054
6.36.4.4 vtss_acl_policer_conf_set()	1054
6.36.4.5 vtss_acl_port_conf_get()	1055
6.36.4.6 vtss_acl_port_conf_set()	1055
6.36.4.7 vtss_acl_port_counter_get()	1056
6.36.4.8 vtss_acl_port_counter_clear()	1056
6.36.4.9 vtss_ace_init()	1056
6.36.4.10 vtss_ace_add()	1057
6.36.4.11 vtss_ace_del()	1057
6.36.4.12 vtss_ace_counter_get()	1058
6.36.4.13 vtss_ace_counter_clear()	1058
6.37 vtss_api/include/vtss_sf4_api.h File Reference	1058
6.37.1 Detailed Description	1059
6.38 vtss_api/include/vtss_sync_api.h File Reference	1059
6.38.1 Detailed Description	1060
6.38.2 Macro Definition Documentation	1060
6.38.2.1 VTSS_SYNCE_CLK_A	1060
6.38.2.2 VTSS_SYNCE_CLK_B	1060
6.38.2.3 VTSS_SYNCE_CLK_MAX	1060
6.38.3 Enumeration Type Documentation	1060
6.38.3.1 vtss_sync_clock_divider_t	1060
6.38.4 Function Documentation	1061
6.38.4.1 vtss_sync_clock_out_set()	1061
6.38.4.2 vtss_sync_clock_out_get()	1061
6.38.4.3 vtss_sync_clock_in_set()	1062
6.38.4.4 vtss_sync_clock_in_get()	1062
6.39 vtss_api/include/vtss_tfi5_api.h File Reference	1062
6.39.1 Detailed Description	1063
6.40 vtss_api/include/vtss_ts_api.h File Reference	1063
6.40.1 Detailed Description	1066

6.40.2 Typedef Documentation . . . . .	1066
6.40.2.1 <code>vtss_ts_alt_clock_mode_t</code> . . . . .	1066
6.40.3 Function Documentation . . . . .	1066
6.40.3.1 <code>vtss_ts_timeofday_set()</code> . . . . .	1066
6.40.3.2 <code>vtss_ts_timeofday_set_delta()</code> . . . . .	1067
6.40.3.3 <code>vtss_ts_timeofday_offset_set()</code> . . . . .	1067
6.40.3.4 <code>vtss_ts_adjtimer_one_sec()</code> . . . . .	1068
6.40.3.5 <code>vtss_ts_ongoing_adjustment()</code> . . . . .	1068
6.40.3.6 <code>vtss_ts_timeofday_get()</code> . . . . .	1068
6.40.3.7 <code>vtss_ts_timeofday_next_pps_get()</code> . . . . .	1069
6.40.3.8 <code>vtss_ts_adjtimer_set()</code> . . . . .	1069
6.40.3.9 <code>vtss_ts_adjtimer_get()</code> . . . . .	1070
6.40.3.10 <code>vtss_ts_freq_offset_get()</code> . . . . .	1070
6.40.3.11 <code>vtss_ts_alt_clock_saved_get()</code> . . . . .	1070
6.40.3.12 <code>vtss_ts_alt_clock_mode_get()</code> . . . . .	1072
6.40.3.13 <code>vtss_ts_alt_clock_mode_set()</code> . . . . .	1072
6.40.3.14 <code>vtss_ts_timeofday_next_pps_set()</code> . . . . .	1073
6.40.3.15 <code>vtss_ts_external_clock_mode_get()</code> . . . . .	1073
6.40.3.16 <code>vtss_ts_external_clock_mode_set()</code> . . . . .	1073
6.40.3.17 <code>vtss_ts_external_clock_saved_get()</code> . . . . .	1074
6.40.3.18 <code>vtss_ts_ingress_latency_set()</code> . . . . .	1074
6.40.3.19 <code>vtss_ts_ingress_latency_get()</code> . . . . .	1075
6.40.3.20 <code>vtss_ts_p2p_delay_set()</code> . . . . .	1075
6.40.3.21 <code>vtss_ts_p2p_delay_get()</code> . . . . .	1075
6.40.3.22 <code>vtss_ts_egress_latency_set()</code> . . . . .	1076
6.40.3.23 <code>vtss_ts_egress_latency_get()</code> . . . . .	1076
6.40.3.24 <code>vtss_ts_delay_asymmetry_set()</code> . . . . .	1077
6.40.3.25 <code>vtss_ts_delay_asymmetry_get()</code> . . . . .	1077
6.40.3.26 <code>vtss_ts_operation_mode_set()</code> . . . . .	1078
6.40.3.27 <code>vtss_ts_operation_mode_get()</code> . . . . .	1078

6.40.3.28 <code>vtss_ts_internal_mode_set()</code>	1078
6.40.3.29 <code>vtss_ts_internal_mode_get()</code>	1079
6.40.3.30 <code>vtss_tx_timestamp_update()</code>	1079
6.40.3.31 <code>vtss_rx_timestamp_get()</code>	1080
6.40.3.32 <code>vtss_rx_timestamp_id_release()</code>	1080
6.40.3.33 <code>vtss_rx_master_timestamp_get()</code>	1080
6.40.3.34 <code>vtss_tx_timestamp_idx_alloc()</code>	1081
6.40.3.35 <code>vtss_timestamp_age()</code>	1081
6.40.3.36 <code>vtss_ts_status_change()</code>	1082
6.41 <code>vtss_api/include/vtss_upi_api.h</code> File Reference	1082
6.41.1 Detailed Description	1082
6.42 <code>vtss_api/include/vtss_wis_api.h</code> File Reference	1082
6.42.1 Detailed Description	1088
6.42.2 Macro Definition Documentation	1088
6.42.2.1 <code>VTSS_EWIS_SEF_EV</code>	1088
6.42.2.2 <code>VTSS_EWIS_FPLM_EV</code>	1088
6.42.2.3 <code>VTSS_EWIS_FAIS_EV</code>	1088
6.42.2.4 <code>VTSS_EWIS_LOF_EV</code>	1089
6.42.2.5 <code>VTSS_EWIS_LOS_EV</code>	1089
6.42.2.6 <code>VTSS_EWIS_RDIL_EV</code>	1089
6.42.2.7 <code>VTSS_EWIS_AISL_EV</code>	1089
6.42.2.8 <code>VTSS_EWIS_LCDCP_EV</code>	1089
6.42.2.9 <code>VTSS_EWIS_PLMP_EV</code>	1090
6.42.2.10 <code>VTSS_EWIS_AISP_EV</code>	1090
6.42.2.11 <code>VTSS_EWIS_LOPP_EV</code>	1090
6.42.2.12 <code>VTSS_EWIS_MODULE_EV</code>	1090
6.42.2.13 <code>VTSS_EWIS_TXLOL_EV</code>	1090
6.42.2.14 <code>VTSS_EWIS_RXLOL_EV</code>	1091
6.42.2.15 <code>VTSS_EWIS_LOPC_EV</code>	1091
6.42.2.16 <code>VTSS_EWIS_UNEQP_EV</code>	1091

6.42.2.17 VTSS_EWIS_FEUNEQP_EV . . . . .	1091
6.42.2.18 VTSS_EWIS_FERDIP_EV . . . . .	1091
6.42.2.19 VTSS_EWIS_REIL_EV . . . . .	1092
6.42.2.20 VTSS_EWIS_REIP_EV . . . . .	1092
6.42.2.21 VTSS_EWIS_HIGH_BER_EV . . . . .	1092
6.42.2.22 VTSS_EWIS_PCS_RECEIVE_FAULT_PEND . . . . .	1092
6.42.2.23 VTSS_EWIS_B1_NZ_EV . . . . .	1092
6.42.2.24 VTSS_EWIS_B2_NZ_EV . . . . .	1093
6.42.2.25 VTSS_EWIS_B3_NZ_EV . . . . .	1093
6.42.2.26 VTSS_EWIS_REIL_NZ_EV . . . . .	1093
6.42.2.27 VTSS_EWIS_REIP_NZ_EV . . . . .	1093
6.42.2.28 VTSS_EWIS_B1_THRESH_EV . . . . .	1093
6.42.2.29 VTSS_EWIS_B2_THRESH_EV . . . . .	1094
6.42.2.30 VTSS_EWIS_B3_THRESH_EV . . . . .	1094
6.42.2.31 VTSS_EWIS_REIL_THRESH_EV . . . . .	1094
6.42.2.32 VTSS_EWIS_REIP_THRESH_EV . . . . .	1094
6.42.3 Typedef Documentation . . . . .	1094
6.42.3.1 vtss_ewis_static_conf_t . . . . .	1094
6.42.3.2 vtss_ewis_event_t . . . . .	1095
6.42.4 Enumeration Type Documentation . . . . .	1095
6.42.4.1 vtss_ewis_tti_mode_t . . . . .	1095
6.42.4.2 vtss_ewis_perf_cntr_mode_t . . . . .	1095
6.42.4.3 vtss_ewis_mode_t . . . . .	1095
6.42.4.4 vtss_ewis_test_pattern_s . . . . .	1096
6.42.4.5 vtss_ewis_prbs31_err_inj_t . . . . .	1096
6.42.5 Function Documentation . . . . .	1096
6.42.5.1 vtss_ewis_event_enable() . . . . .	1097
6.42.5.2 vtss_ewis_event_poll() . . . . .	1097
6.42.5.3 vtss_ewis_event_poll_without_mask() . . . . .	1098
6.42.5.4 vtss_ewis_event_force() . . . . .	1098

6.42.5.5 vtss_ewis_static_conf_get()	1099
6.42.5.6 vtss_ewis_force_conf_set()	1099
6.42.5.7 vtss_ewis_force_conf_get()	1100
6.42.5.8 vtss_ewis_tx_oh_set()	1100
6.42.5.9 vtss_ewis_tx_oh_get()	1100
6.42.5.10 vtss_ewis_tx_oh_passthru_set()	1101
6.42.5.11 vtss_ewis_tx_oh_passthru_get()	1101
6.42.5.12 vtss_ewis_mode_set()	1102
6.42.5.13 vtss_ewis_mode_get()	1102
6.42.5.14 vtss_ewis_reset()	1103
6.42.5.15 vtss_ewis_cons_act_set()	1103
6.42.5.16 vtss_ewis_cons_act_get()	1104
6.42.5.17 vtss_ewis_section_txти_set()	1104
6.42.5.18 vtss_ewis_section_txти_get()	1104
6.42.5.19 vtss_ewis_exp_sl_set()	1105
6.42.5.20 vtss_ewis_path_txти_set()	1105
6.42.5.21 vtss_ewis_path_txти_get()	1106
6.42.5.22 vtss_ewis_test_mode_set()	1106
6.42.5.23 vtss_ewis_test_mode_get()	1107
6.42.5.24 vtss_ewis_prbs31_err_inj_set()	1107
6.42.5.25 vtss_ewis_test_counter_get()	1108
6.42.5.26 vtss_ewis_defects_get()	1108
6.42.5.27 vtss_ewis_status_get()	1109
6.42.5.28 vtss_ewis_section_acti_get()	1109
6.42.5.29 vtss_ewis_path_acti_get()	1110
6.42.5.30 vtss_ewis_counter_get()	1110
6.42.5.31 vtss_ewis_perf_get()	1110
6.42.5.32 vtss_ewis_counter_threshold_set()	1111
6.42.5.33 vtss_ewis_counter_threshold_get()	1111
6.42.5.34 vtss_ewis_perf_mode_set()	1112
6.42.5.35 vtss_ewis_perf_mode_get()	1112
6.43 vtss_api/include/vtss_xaui_api.h File Reference	1113
6.43.1 Detailed Description	1113
6.44 vtss_api/include/vtss_xfi_api.h File Reference	1113
6.44.1 Detailed Description	1113



# Chapter 1

## Layer 2

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

### 1.1 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.

## 1.2 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.

## 1.3 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.

## 1.4 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.

## 1.5 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.

## 1.6 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.

## 1.7 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.

## 1.8 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.

## 1.9 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.

## 1.10 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.

## 1.11 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.

## 1.12 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.

## 1.13 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.

## 1.14 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.

## 1.15 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.

## 1.16 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.

## 1.17 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.

## 1.18 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.

## 1.19 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.

## 1.20 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 2

## Security

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

### 2.1 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.

### 2.2 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.

#### 2.2.1 Access Control Entry

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

- [vtss\\_ace\\_init\(\)](#) is used to initialize an ACE to default values.
- [vtss\\_ace\\_add\(\)](#) is used to add or modify an ACE.
- [vtss\\_ace\\_del\(\)](#) is used to delete an ACE.
- [vtss\\_ace\\_counter\\_get\(\)](#) is used to get the hit counter of an ACE.

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

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

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

## 2.2.2 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.

## 2.2.3 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 3

## Data Structure Index

### 3.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">ib_par_cfg</a>	Generalized data structure for IB parameters . . . . .	25
<a href="#">port_custom_conf_t</a>	Port configuration . . . . .	26
<a href="#">serdes_fields_t</a>	Serdes fields . . . . .	29
<a href="#">tag_vtss_fdma_list</a>	Software DCB structure . . . . .	30
<a href="#">vtss_ace_frame_arp_t</a>	Frame data for VTSS_ACE_TYPE_ARP . . . . .	33
<a href="#">vtss_ace_frame_etype_t</a>	Frame data for VTSS_ACE_TYPEETYPE . . . . .	36
<a href="#">vtss_ace_frame_ipv4_t</a>	Frame data for VTSS_ACE_TYPE_IPV4 . . . . .	38
<a href="#">vtss_ace_frame_ipv6_t</a>	Frame data for VTSS_ACE_TYPE_IPV6 . . . . .	43
<a href="#">vtss_ace_frame_llc_t</a>	Frame data for VTSS_ACE_TYPE_LLCC . . . . .	47
<a href="#">vtss_ace_frame_snap_t</a>	Frame data for VTSS_ACE_TYPE_SNAP . . . . .	48
<a href="#">vtss_ace_ptp_t</a>	PTP header filtering . . . . .	49
<a href="#">vtss_ace_sip_smac_t</a>	SIP/SMAC filtering . . . . .	50
<a href="#">vtss_ace_t</a>	Access Control Entry . . . . .	51
<a href="#">vtss_ace_vlan_t</a>	ACE VLAN information . . . . .	55
<a href="#">vtss_acl_action_t</a>	ACL Action . . . . .	57
<a href="#">vtss_acl_policer_conf_t</a>	ACL policer configuration . . . . .	59
<a href="#">vtss_acl_port_conf_t</a>	ACL port configuration . . . . .	61
<a href="#">vtss_acl_ptp_action_conf_t</a>	ACL PTP action configuration . . . . .	61

<code>vtss_aggr_mode_t</code>	Aggregation traffic distribution mode . . . . .	63
<code>vtss_aneg_t</code>	Auto negotiation struct . . . . .	64
<code>vtss_api_lock_t</code>	API lock structure . . . . .	65
<code>vtss_basic_counters_t</code>	Basic counters structure . . . . .	67
<code>vtss_chip_id_t</code>	Chip ID . . . . .	68
<code>vtss_debug_info_t</code>	Debug information structure . . . . .	69
<code>vtss_debug_lock_t</code>	API debug lock structure . . . . .	71
<code>vtss_dgroup_port_conf_t</code>	Destination group port configuration . . . . .	72
<code>vtss_dlb_policer_conf_t</code>	Dual leaky buckets policer configuration . . . . .	72
<code>vtss_eee_port_conf_t</code>	EEE port configuration . . . . .	75
<code>vtss_eee_port_counter_t</code>	EEE port counters (JR only) . . . . .	76
<code>vtss_eee_port_state_t</code>	EEE port state (JR only) . . . . .	78
<code>vtss_eps_port_conf_t</code>	Port protection configuration . . . . .	79
<code>vtss_ewis_aisl_cons_act_s</code>	EWIS AIS-L consequent actions . . . . .	80
<code>vtss_ewis_conf_s</code>	EWIS configuration primitives, used to hold all the configuration parameters in the internal state in the API . . . . .	81
<code>vtss_ewis_cons_act_s</code>	EWIS consequent actions . . . . .	84
<code>vtss_ewis_counter_s</code>	EWIS performance counters. These counters are free running counters that wraps to zero . . . . .	85
<code>vtss_ewis_counter_threshold_s</code>	EWIS performance counter thresholds . . . . .	87
<code>vtss_ewis_defects_s</code>	EWIS defects . . . . .	88
<code>vtss_ewis_fault_cons_act_s</code>	EWIS fault mask configuration, i.e set up which defects trigger the Fault condition . . . . .	92
<code>vtss_ewis_force_mode_s</code>	EWIS force modes . . . . .	95
<code>vtss_ewis_line_force_mode_s</code>	EWIS line force mode . . . . .	96
<code>vtss_ewis_line_tx_force_mode_s</code>	EWIS line TX force mode . . . . .	97
<code>vtss_ewis_path_force_mode_s</code>	EWIS path force modes . . . . .	98
<code>vtss_ewis_perf_mode_s</code>	EWIS Mode(Bit/Block) for the Performance Monitoring Counters . . . . .	99
<code>vtss_ewis_perf_s</code>	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 . . . . .	101
<code>vtss_ewis_rdil_cons_act_s</code>	EWIS RDI-L consequent actions . . . . .	102

<a href="#">vtss_ewis_sl_conf_s</a>	Signal label configuration . . . . .	104
<a href="#">vtss_ewis_static_conf_s</a>	EWIS static configuration data, . . . . .	104
<a href="#">vtss_ewis_status_s</a>	EWIS status . . . . .	108
<a href="#">vtss_ewis_test_conf_s</a>	EWIS test configuration . . . . .	109
<a href="#">vtss_ewis_test_status_s</a>	EWIS test status . . . . .	110
<a href="#">vtss_ewis_tti_s</a>	Trail Trace Identifier type . . . . .	111
<a href="#">vtss_ewis_tx_oh_s</a>	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 . . . . .	112
<a href="#">vtss_ewis_tx_passthru_s</a>	EWIS overhead passthru configuration . . . . .	116
<a href="#">vtss_fan_conf_t</a>	Fan specifications . . . . .	120
<a href="#">vtss_fdma_cfg_t</a>	FDMA configuration structure . . . . .	121
<a href="#">vtss_fdma_ch_cfg_t</a>	Channel configuration structure . . . . .	126
<a href="#">vtss_fdma_throttle_cfg_t</a>	. . . . .	130
<a href="#">vtss_fdma_tx_info_t</a>	FDMA Injection Properties . . . . .	131
<a href="#">vtss_gpio_10g_gpio_mode_t</a>	GPIO configured mode . . . . .	133
<a href="#">vtss_init_conf_t</a>	Initialization configuration . . . . .	135
<a href="#">vtss_inst_create_t</a>	Create structure . . . . .	139
<a href="#">vtss_intr_t</a>	Interrupt source structure . . . . .	140
<a href="#">vtss_ip_addr_t</a>	Either an IPv4 or IPv6 address . . . . .	141
<a href="#">vtss_ip_network_t</a>	IPv6 network . . . . .	142
<a href="#">vtss_ipv4_network_t</a>	IPv4 network . . . . .	143
<a href="#">vtss_ipv4_uc_t</a>	IPv4 unicast routing entry . . . . .	144
<a href="#">vtss_ipv6_network_t</a>	IPv6 network . . . . .	145
<a href="#">vtss_ipv6_t</a>	IPv6 address/mask . . . . .	146
<a href="#">vtss_ipv6_uc_t</a>	IPv6 routing entry . . . . .	147
<a href="#">vtss_irq_conf_t</a>	Interrupt configuration options . . . . .	147
<a href="#">vtss_irq_status_t</a>	Interrupt status structure . . . . .	148
<a href="#">vtss_l3_counters_t</a>	Routing interface statics counter . . . . .	150
<a href="#">vtss_lcpll_status_t</a>	Structure for Get PHY LC-PLL status . . . . .	152

vtss_learn_mode_t	Learning mode . . . . .	154
vtss_mac_t	MAC Address . . . . .	155
vtss_mac_table_entry_t	MAC address entry . . . . .	156
vtss_mac_table_status_t	MAC address table status . . . . .	159
vtss_mirror_conf_t	Mirror configuration . . . . .	160
vtss_mtimer_t	Timer structure . . . . .	162
vtss_npi_conf_t	NPI configuration . . . . .	163
vtss_os_timestamp_t	. . . . .	164
vtss_packet_dma_conf_t	. . . . .	165
vtss_packet_frame_info_t	Information about frame . . . . .	166
vtss_packet_port_filter_t	Packet information for each port . . . . .	167
vtss_packet_port_info_t	Port info structure . . . . .	168
vtss_packet_rx_conf_t	CPU Rx configuration . . . . .	170
vtss_packet_rx_header_t	System frame header describing received frame . . . . .	171
vtss_packet_rx_info_t	Decoded extraction header properties . . . . .	173
vtss_packet_rx_meta_t	Input structure to <code>vtss_packet_rx_hdr_decode()</code> . . . . .	183
vtss_packet_rx_port_conf_t	Packet registration per port . . . . .	187
vtss_packet_rx_queue_conf_t	CPU Rx queue configuration . . . . .	188
vtss_packet_rx_queue_map_t	CPU Rx queue map . . . . .	189
vtss_packet_rx_queue_npi_conf_t	CPU Rx queue NPI configuration . . . . .	191
vtss_packet_rx_reg_t	CPU Rx packet registration . . . . .	192
vtss_packet_tx_ifh_t	Compiled Tx Frame Header . . . . .	194
vtss_packet_tx_info_t	Injection Properties . . . . .	195
vtss_phy_10g_apc_conf_t	10G Phy APC configuration . . . . .	206
vtss_phy_10g_apc_status_t	10G Phy APC status . . . . .	207
vtss_phy_10g_auto_failover_conf_t	10G PHY Automatic Failover configuration . . . . .	208
vtss_phy_10g_base_kr_autoneg_t	10G Phy Base KR Autoneg config . . . . .	210
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 . . . . .	212
vtss_phy_10g_base_kr_id_adv_abil_t	10G Phy Base Link Advertisement capability config . . . . .	214
vtss_phy_10g_base_kr_status_t	10G Phy Base KR Training & Autoneg status . . . . .	215

<a href="#">vtss_phy_10g_base_kr_train_aneg_t</a>	10G Phy Base KR Training & Autoneg config . . . . .	216
<a href="#">vtss_phy_10g_base_kr_training_t</a>	10G Phy Base KR Training config . . . . .	218
<a href="#">vtss_phy_10g_ckout_conf_t</a>	10G Phy CKOUT config data . . . . .	219
<a href="#">vtss_phy_10g_clause_37_adv_t</a>	Advertisement control data for Clause 37 aneg . . . . .	221
<a href="#">vtss_phy_10g_clause_37_cmn_status_t</a>	Clause 37 Auto-negotiation status for line and host . . . . .	223
<a href="#">vtss_phy_10g_clause_37_control_t</a>	Clause 37 control struct . . . . .	224
<a href="#">vtss_phy_10g_clause_37_status_t</a>	Clause 37 Auto-negotiation status . . . . .	226
<a href="#">vtss_phy_10g_clk_src_t</a>	10G Phy CLOCK Source Selection . . . . .	227
<a href="#">vtss_phy_10g_cnt_t</a>	10G Phy Sublayer counters . . . . .	228
<a href="#">vtss_phy_10g_fifo_sync_t</a>	10G OOS workaround options . . . . .	229
<a href="#">vtss_phy_10g_fw_status_t</a>	Firmware status . . . . .	230
<a href="#">vtss_phy_10g_host_clk_conf_t</a>	10G Phy Host clock config data . . . . .	231
<a href="#">vtss_phy_10g_ib_conf_t</a>	10G Phy IB configuration . . . . .	232
<a href="#">vtss_phy_10g_ib_status_t</a>	10G Phy IB configuration . . . . .	237
<a href="#">vtss_phy_10g_ib_storage_t</a>	VSCOPE fast scan storage . . . . .	238
<a href="#">vtss_phy_10g_id_t</a>	10G Phy part number and revision . . . . .	239
<a href="#">vtss_phy_10g_init_parm_t</a>	10G Phy Initialization configuration . . . . .	241
<a href="#">vtss_phy_10g_jitter_conf_t</a>	10G Phy Optimisation of jitter performance . . . . .	242
<a href="#">vtss_phy_10g_kr_status_aneg_t</a>	10G Phy Base KR Autoneg status . . . . .	243
<a href="#">vtss_phy_10g_kr_status_fec_t</a>	10G Phy Base KR FEC status . . . . .	245
<a href="#">vtss_phy_10g_kr_status_train_t</a>	10G Phy Base KR Training status . . . . .	246
<a href="#">vtss_phy_10g_lane_sync_conf_t</a>	10G Phy Lane SYNC Configuration . . . . .	248
<a href="#">vtss_phy_10g_line_clk_conf_t</a>	10G Phy Line clock config data . . . . .	249
<a href="#">vtss_phy_10g_loopback_t</a>	10G Phy system and network loopbacks . . . . .	250
<a href="#">vtss_phy_10g_mode_t</a>	10G Phy operating mode . . . . .	251
<a href="#">vtss_phy_10g_ob_status_t</a>	10G Phy OB status . . . . .	264
<a href="#">vtss_phy_10g_pcs_prbs_gen_conf_t</a>	. . . . .	266
<a href="#">vtss_phy_10g_pcs_prbs_mon_conf_t</a>	. . . . .	267
<a href="#">vtss_phy_10g_pkt_gen_conf_t</a>	10G PHY Packet generator configuration . . . . .	268
<a href="#">vtss_phy_10g_pkt_mon_conf_t</a>	10G PHY Packet Monitor configuration . . . . .	271

vtss_phy_10g_polarity_inv_t	10G Phy Polarity inversion	273
vtss_phy_10g_prbs_gen_conf_t	10G PHY prbs monitor Configuration	275
vtss_phy_10g_prbs_mon_conf_t	10G PHY RXCKOUT config data	276
vtss_phy_10g_rxckout_conf_t	10G Phy SERDES status	281
vtss_phy_10g_sckout_conf_t	10G Phy SCKOUT config data	282
vtss_phy_10g_serdes_status_t	10G Phy link and fault status for all sublayers	284
vtss_phy_10g_srefclk_mode_t	10G Phy srefclk config data	291
vtss_phy_10g_status_t	10G PHY timestamp value array(holder)	291
vtss_phy_10g_txckout_conf_t	10G TXCKOUT config data	295
vtss_phy_10g_vscope_conf_t	VSCOPE scan configuration	296
vtss_phy_10g_vscope_scan_conf_t	SPI daisy chain configuration	297
vtss_phy_aneg_t	PHY auto negotiation advertisement	299
vtss_phy_clock_conf_t	PHY clock configuration	301
vtss_phy_conf_1g_t	PHY 1G configuration	303
vtss_phy_conf_t	PHY configuration	304
vtss_phy_daisy_chain_conf_t	EEE configuration	305
vtss_phy_eee_conf_t	Enhanced LED control	308
vtss_phy_forced_t	PHY forced mode configuration	309
vtss_phy_led_mode_select_t	LED model selection	310
vtss_phy_loopback_t	1G Phy loopbacks	311
vtss_phy_ltc_freq_synth_s	Frequency systhesis pulse configuration	312
vtss_phy_mac_serdes_pcs_ctrl_t	PHY MAC SerDes PCS Control, Reg16E3	313
vtss_phy_media_serdes_pcs_ctrl_t	PHY MEDIA SerDes PCS Control, Reg23E3	316
vtss_phy_pcs_cnt_t	10G Phy PCS counters	317
vtss_phy_power_conf_t	PHY power configuration	320
vtss_phy_power_status_t	PHY power status	322
vtss_phy_reset_conf_t	PHY reset structure	324

<code>vtss_phy_rgmii_conf_t</code>	PHY RGMII configuration . . . . .	327
<code>vtss_phy_statistic_t</code>	Phy statistic information . . . . .	328
<code>vtss_phy_status_1g_t</code>	PHY 1G status . . . . .	330
<code>vtss_phy_tbi_conf_t</code>	PHY TBI configuration . . . . .	331
<code>vtss_phy_timestamp_t</code>	PHY timestamp in seconds and nanoseconds (10 bytes Timestamp) . . . . .	332
<code>vtss_phy_ts_ach_conf_t</code>	Analyzer ACH comparator configuration options . . . . .	333
<code>vtss_phy_ts_alt_clock_mode_s</code>	Parameter for setting the alternative clock mode . . . . .	335
<code>vtss_phy_ts_eng_init_conf_t</code>	Defines the basic engine parameters passed to the engine_init_conf_get() function . . . . .	336
<code>vtss_phy_ts_engine_action_t</code>	Engine Action configuration options . . . . .	338
<code>vtss_phy_ts_engine_flow_conf_t</code>	Analyzer flow configuration options . . . . .	340
<code>vtss_phy_ts_eth_conf_t</code>	Analyzer Ethernet comparator configuration options . . . . .	342
<code>vtss_phy_ts_fifo_conf_t</code>	Defines the params for FIFO SYNC function . . . . .	348
<code>vtss_phy_ts_fifo_sig_t</code>	Tx TSFIFO entry signature . . . . .	349
<code>vtss_phy_ts_gen_conf_t</code>	Analyzer Generic data configuration options using IP comparator . . . . .	351
<code>vtss_phy_ts_generic_action_t</code>	Generic Action configuration option . . . . .	353
<code>vtss_phy_ts_generic_flow_conf_t</code>	Generic engine flow configuration options . . . . .	355
<code>vtss_phy_ts_ietf_mpls_ach_oam_conf_t</code>	Analyzer OAM comparator, IETF MPLS ACH OAM Packet format configuration options . . . . .	356
<code>vtss_phy_ts_init_conf_t</code>	Defines the initial parameters to be passed to init function . . . . .	357
<code>vtss_phy_ts_ip_conf_t</code>	Analyzer IP comparator configuration options . . . . .	361
<code>vtss_phy_ts_mpls_conf_t</code>	Analyzer MPLS comparator configuration options . . . . .	365
<code>vtss_phy_ts_mpls_lvl_rng_t</code>	MPLS level range . . . . .	369
<code>vtss_phy_ts_nphase_status_t</code>	N-phase status . . . . .	370
<code>vtss_phy_ts_oam_engine_action_t</code>	OAM Action configuration options . . . . .	371
<code>vtss_phy_ts_oam_engine_flow_conf_t</code>	OAM engine flow configuration options . . . . .	373
<code>vtss_phy_ts_pps_config_s</code>	PPS Configuration . . . . .	374
<code>vtss_phy_ts_ptp_conf_t</code>	Analyzer PTP comparator configuration options . . . . .	376
<code>vtss_phy_ts_ptp_engine_action_t</code>	Analyzer PTP action configuration options . . . . .	378
<code>vtss_phy_ts_ptp_engine_flow_conf_t</code>	PTP engine flow configuration options . . . . .	379
<code>vtss_phy_ts_sertod_conf_t</code>	PHY timestamp in seconds and nanoseconds (10 bytes Timestamp) . . . . .	381

<code>vtss_phy_ts_stats_t</code>	Timestamping Statistics . . . . .	382
<code>vtss_phy_ts_y1731_oam_conf_t</code>	Analyzer OAM comparator, Y.1731 OAM Packet format configuration options . . . . .	385
<code>vtss_phy_type_t</code>	Phy type information . . . . .	387
<code>vtss_phy_veriphy_result_t</code>	VeriPHY result . . . . .	389
<code>vtss_phy_wol_conf_t</code>	Structure for Get/Set Wake-On-LAN configuration . . . . .	390
<code>vtss_pi_conf_t</code>	PI configuration . . . . .	391
<code>vtss_policer_ext_t</code>	Policer Extensions . . . . .	392
<code>vtss_policer_t</code>	Policer . . . . .	396
<code>vtss_port_bridge_counters_t</code>	Port bridge counter structure (RFC 4188) . . . . .	396
<code>vtss_port_clause_37_adv_t</code>	Advertisement control data for Clause 37 aneg . . . . .	397
<code>vtss_port_clause_37_control_t</code>	Auto-negotiation control parameter struct . . . . .	399
<code>vtss_port_conf_t</code>	Port configuration structure . . . . .	400
<code>vtss_port_counters_t</code>	Port counter structure . . . . .	404
<code>vtss_port_ethernet_like_counters_t</code>	Ethernet-like Interface counter structure (RFC 3635) . . . . .	406
<code>vtss_port_flow_control_conf_t</code>	Flow control setup . . . . .	409
<code>vtss_port_frame_gaps_t</code>	Inter frame gap structure . . . . .	411
<code>vtss_port_if_group_counters_t</code>	Interfaces Group counter structure (RFC 2863) . . . . .	412
<code>vtss_port_ifh_t</code>	Port Internal Frame Header structure . . . . .	415
<code>vtss_port_map_t</code>	Port map structure . . . . .	417
<code>vtss_port_proprietary_counters_t</code>	Port proprietary counter structure . . . . .	418
<code>vtss_port_rmon_counters_t</code>	RMON counter structure (RFC 2819) . . . . .	419
<code>vtss_port_serdes_conf_t</code>	SFI Serdes configuration . . . . .	426
<code>vtss_port_sgmii_aneg_t</code>	Advertisement control data for SGMII aneg . . . . .	427
<code>vtss_port_status_t</code>	Port status parameter struct . . . . .	429
<code>vtss_qce_action_t</code>	QCE action . . . . .	432
<code>vtss_qce_frame_etype_t</code>	Frame data for VTSS_QCE_TYPE_ETYPE . . . . .	435
<code>vtss_qce_frame_ipv4_t</code>	Frame data for VTSS_QCE_TYPE_IPV4 . . . . .	435
<code>vtss_qce_frame_ipv6_t</code>	Frame data for VTSS_QCE_TYPE_IPV6 . . . . .	437
<code>vtss_qce_frame_llc_t</code>	Frame data for VTSS_QCE_TYPE_LLCC . . . . .	439

<code>vtss_qce_frame_snap_t</code>	Frame data for VTSS_QCE_TYPE_SNAP . . . . .	440
<code>vtss_qce_key_t</code>	QCE key . . . . .	441
<code>vtss_qce_mac_t</code>	QCE MAC information . . . . .	443
<code>vtss_qce_t</code>	QoS Control Entry . . . . .	445
<code>vtss_qce_tag_t</code>	QCE tag information . . . . .	446
<code>vtss_qos_conf_t</code>	All parameters below are defined per chip . . . . .	448
<code>vtss_qos_port_conf_t</code>	QoS setup per port . . . . .	453
<code>vtss_rcpll_status_t</code>	Structure for Get PHY RC-PLL status . . . . .	458
<code>vtss_red_v2_t</code>	Random Early Detection configuration struct version 2 (per queue, per dpl - switch global) . . . . .	460
<code>vtss_restart_status_t</code>	Restart status . . . . .	461
<code>vtss_routing_entry_t</code>	Routing entry . . . . .	462
<code>vtss_secure_on_passwd_t</code>	Structure for Wake-On-LAN Secure-On Password . . . . .	464
<code>vtss_serdes_macro_conf_t</code>	Serdes macro configuration . . . . .	465
<code>vtss_sflow_port_conf_t</code>	SFlow configuration structure . . . . .	466
<code>vtss_sgpi_conf_t</code>	SGPIO configuration for a group . . . . .	467
<code>vtss_sgpi_port_conf_t</code>	SGPIO port configuration . . . . .	468
<code>vtss_sgpi_port_data_t</code>	SGPIO read data for a port . . . . .	469
<code>vtss_shaper_t</code>	Shaper . . . . .	470
<code>vtss_sublayer_status_t</code>	10G Phy link and fault status . . . . .	471
<code>vtss_sync_clock_in_t</code>	Struct containing configuration selecting the recovered input clock port, to be delivered to a selected output clock port . . . . .	473
<code>vtss_sync_clock_out_t</code>	Struct containing configuration for a recovered clock output port . . . . .	474
<code>vtss_tci_t</code>	Tag Control Information (according to IEEE 802.1Q) . . . . .	475
<code>vtss_timeofday_t</code>	Time of day structure . . . . .	476
<code>vtss_timestamp_t</code>	Time stamp in seconds and nanoseconds . . . . .	477
<code>vtss_trace_conf_t</code>	Trace group configuration . . . . .	478
<code>vtss_ts_alt_clock_mode_t</code>	Parameter for setting the alternative clock mode . . . . .	479
<code>vtss_ts_ext_clock_mode_t</code>	External clock output configuration . . . . .	480
<code>vtss_ts_id_t</code>	Timestamp identifier . . . . .	482

<a href="#">vtss_ts_internal_mode_t</a>	Hardware timestamping format mode for internal ports . . . . .	482
<a href="#">vtss_ts_operation_mode_t</a>	Timestamp operation . . . . .	483
<a href="#">vtss_ts_timestamp_alloc_t</a>	Timestamp allocation . . . . .	484
<a href="#">vtss_ts_timestamp_t</a>	Timestamp structure . . . . .	485
<a href="#">vtss_vcap_ip_t</a>	VCAP IPv4 address value and mask . . . . .	486
<a href="#">vtss_vcap_u128_t</a>	VCAP 128 bit value and mask . . . . .	487
<a href="#">vtss_vcap_u16_t</a>	VCAP 16 bit value and mask . . . . .	488
<a href="#">vtss_vcap_u24_t</a>	VCAP 24 bit value and mask . . . . .	489
<a href="#">vtss_vcap_u32_t</a>	VCAP 32 bit value and mask . . . . .	490
<a href="#">vtss_vcap_u40_t</a>	VCAP 40 bit value and mask . . . . .	491
<a href="#">vtss_vcap_u48_t</a>	VCAP 48 bit value and mask . . . . .	492
<a href="#">vtss_vcap_u8_t</a>	VCAP 8 bit value and mask . . . . .	493
<a href="#">vtss_vcap_udp_tcp_t</a>	VCAP UDP/TCP port range . . . . .	494
<a href="#">vtss_vcap_vid_t</a>	VCAP VLAN ID value and mask . . . . .	495
<a href="#">vtss_vcap_vr_t</a>	VCAP universal value or range . . . . .	496
<a href="#">vtss_vce_action_t</a>	VCE Action . . . . .	499
<a href="#">vtss_vce_frame_etype_t</a>	Frame data for VTSS_VCE_TYPE_ETYPE . . . . .	499
<a href="#">vtss_vce_frame_ipv4_t</a>	Frame data for VTSS_VCE_TYPE_IPV4 . . . . .	500
<a href="#">vtss_vce_frame_ipv6_t</a>	Frame data for VTSS_VCE_TYPE_IPV6 . . . . .	502
<a href="#">vtss_vce_frame_llc_t</a>	Frame data for VTSS_VCE_TYPE_LLCC . . . . .	504
<a href="#">vtss_vce_frame_snap_t</a>	Frame data for VTSS_VCE_TYPE_SNAP . . . . .	504
<a href="#">vtss_vce_key_t</a>	VCE Key . . . . .	505
<a href="#">vtss_vce_mac_t</a>	VCE MAC header information . . . . .	508
<a href="#">vtss_vce_t</a>	VLAN Control Entry . . . . .	509
<a href="#">vtss_vce_tag_t</a>	VCE tag information . . . . .	510
<a href="#">vtss_vcl_port_conf_t</a>	VCL port configuration . . . . .	512
<a href="#">vtss_vid_mac_t</a>	MAC Address in specific VLAN . . . . .	512
<a href="#">vtss_vlan_conf_t</a>	VLAN configuration . . . . .	513
<a href="#">vtss_vlan_port_conf_t</a>	VLAN port configuration . . . . .	514

<code>vtss_vlan_tag_t</code>	515
<code>vtss_vlan_trans_grp2vlan_conf_t</code>	
VLAN translation group-to-VLAN configuration	517
<code>vtss_vlan_trans_port2grp_conf_t</code>	
VLAN translation port-to-group configuration	518
<code>vtss_vlan_vid_conf_t</code>	
VLAN ID configuration	519
<code>vtss_vstax_conf_t</code>	
VStaX configuration for switch	520
<code>vtss_vstax_glag_entry_t</code>	
GLAG info	522
<code>vtss_vstax_port_conf_t</code>	
VStaX setup for port	522
<code>vtss_vstax_route_entry_t</code>	
UPSID Route Entry	523
<code>vtss_vstax_route_table_t</code>	
UPSID Route Table	524
<code>vtss_vstax_rx_header_t</code>	
VStaX frame header used for reception	525
<code>vtss_vstax_tx_header_t</code>	
VStaX frame header used for transmission	527
<code>vtss_wol_mac_addr_t</code>	
Structure for Wake-On-LAN MAC Address	530



# Chapter 4

## File Index

### 4.1 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 . . . . .	605
vtss_api/include/ <a href="#">vtss_afi_api.h</a>	
AFI API . . . . .	605
vtss_api/include/ <a href="#">vtss_aneg_api.h</a>	
ANEG API . . . . .	605
vtss_api/include/ <a href="#">vtss_api.h</a>	
Vitesse API main header file . . . . .	606
vtss_api/include/ <a href="#">vtss_evc_api.h</a>	
EVC API . . . . .	606
vtss_api/include/ <a href="#">vtss_fdma_api.h</a>	
Frame DMA API . . . . .	608
vtss_api/include/ <a href="#">vtss_gfp_api.h</a>	
GFP API . . . . .	622
vtss_api/include/ <a href="#">vtss_hqos_api.h</a>	
HQoS API . . . . .	622
vtss_api/include/ <a href="#">vtss_i2c_api.h</a>	
I2C API . . . . .	623
vtss_api/include/ <a href="#">vtss_init_api.h</a>	
Initialization API . . . . .	623
vtss_api/include/ <a href="#">vtss_l2_api.h</a>	
Layer 2 API . . . . .	637
vtss_api/include/ <a href="#">vtss_l3_api.h</a>	
L3 routing API . . . . .	702
vtss_api/include/ <a href="#">vtss_mac10g_api.h</a>	
MAC10G API . . . . .	703
vtss_api/include/ <a href="#">vtss_macsec_api.h</a>	
MACsec API . . . . .	??
vtss_api/include/ <a href="#">vtss_misc_api.h</a>	
Miscellaneous API . . . . .	703
vtss_api/include/ <a href="#">vtss_mpls_api.h</a>	
MPLS API . . . . .	736
vtss_api/include/ <a href="#">vtss_oam_api.h</a>	
OAM API . . . . .	737
vtss_api/include/ <a href="#">vtss_oha_api.h</a>	
OHA API . . . . .	737

vtss_api/include/vtss_os.h	
OS Layer API . . . . .	737
vtss_api/include/vtss_os_custom.h	
OS custom header file . . . . .	737
vtss_api/include/vtss_os_ecos.h	
ECos OS API . . . . .	742
vtss_api/include/vtss_os_linux.h	
Linux OS API . . . . .	752
vtss_api/include/vtss_otn_api.h	
OTN API . . . . .	759
vtss_api/include/vtss_packet_api.h	
Packet API . . . . .	759
vtss_api/include/vtss_pcs_10gbase_r_api.h	
PCS_10BASE_R API . . . . .	782
vtss_api/include/vtss_phy_10g_api.h	
10G PHY API . . . . .	782
vtss_api/include/vtss_phy_api.h	
PHY API . . . . .	876
vtss_api/include/vtss_phy_ts_api.h	
PHY TimeStamping API . . . . .	952
vtss_api/include/vtss_port_api.h	
Port API . . . . .	1022
vtss_api/include/vtss_qos_api.h	
QoS API . . . . .	1038
vtss_api/include/vtss_rab_api.h	
RAB API . . . . .	1047
vtss_api/include/vtss_security_api.h	
Security API . . . . .	1048
vtss_api/include/vtss_sfi4_api.h	
SFI4 API . . . . .	1058
vtss_api/include/vtss_sync_api.h	
Synchronization API . . . . .	1059
vtss_api/include/vtss_tfi5_api.h	
TFI5 API . . . . .	1062
vtss_api/include/vtss_ts_api.h	
TimeStamping API . . . . .	1063
vtss_api/include/vtss_upi_api.h	
Define UPI API interface . . . . .	1082
vtss_api/include/vtss_wis_api.h	
EWIS layer API . . . . .	1082
vtss_api/include/vtss_xaui_api.h	
XAUI API . . . . .	1113
vtss_api/include/vtss_xfi_api.h	
XFI API . . . . .	1113
vtss_api/include/vtss/api/l2_types.h	
Layer 2 Public API Header for l2 . . . . .	531
vtss_api/include/vtss/api/options.h	
Features and options . . . . .	532
vtss_api/include/vtss/api/phy.h	
PHY Public API Header . . . . .	554
vtss_api/include/vtss/api/port.h	
Port Public API Header . . . . .	556
vtss_api/include/vtss/api/types.h	
Generic types API . . . . .	569

## Chapter 5

# Data Structure Documentation

### 5.1 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

#### 5.1.1 Detailed Description

Generalized data structure for IB parameters.

Definition at line 207 of file vtss\_phy\_10g\_api.h.

#### 5.1.2 Field Documentation

##### 5.1.2.1 value

```
u16 ib_par_cfg::value
```

value to be configured

Definition at line 208 of file vtss\_phy\_10g\_api.h.

### 5.1.2.2 min

`u16 ib_par_cfg::min`

Minimum value

Definition at line 209 of file vtss\_phy\_10g\_api.h.

### 5.1.2.3 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

## 5.2 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`

### 5.2.1 Detailed Description

Port configuration.

Definition at line 269 of file port.h.

## 5.2.2 Field Documentation

### 5.2.2.1 enable

`BOOL port_custom_conf_t::enable`

Admin enable/disable

Definition at line 270 of file port.h.

### 5.2.2.2 autoneg

`BOOL port_custom_conf_t::autoneg`

Auto negotiation

Definition at line 271 of file port.h.

### 5.2.2.3 fdx

`BOOL port_custom_conf_t::fdx`

Forced duplex mode

Definition at line 272 of file port.h.

### 5.2.2.4 flow\_control

`BOOL port_custom_conf_t::flow_control`

Flow control (Standard 802.3x)

Definition at line 273 of file port.h.

### 5.2.2.5 pfc

`BOOL port_custom_conf_t::pfc[VTSS_PRIOS]`

Priority Flow control (802.1Qbb)

Definition at line 275 of file port.h.

### 5.2.2.6 speed

```
vtss_port_speed_t port_custom_conf_t::speed
```

Forced port speed

Definition at line 277 of file port.h.

### 5.2.2.7 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.

### 5.2.2.8 max\_length

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

Max frame length

Definition at line 279 of file port.h.

### 5.2.2.9 power\_mode

```
vtss_phy_power_mode_t port_custom_conf_t::power_mode
```

PHY power mode

Definition at line 281 of file port.h.

### 5.2.2.10 exc\_col\_cont

```
BOOL port_custom_conf_t::exc_col_cont
```

Excessive collision continuation

Definition at line 283 of file port.h.

#### 5.2.2.11 adv\_dis

`u8 port_custom_conf_t::adv_dis`

Auto neg advertisement disable

Definition at line 284 of file port.h.

#### 5.2.2.12 max\_tags

`u8 port_custom_conf_t::max_tags`

Maximum number of tags

Definition at line 285 of file port.h.

#### 5.2.2.13 oper\_up

`BOOL port_custom_conf_t::oper_up`

Force operational state up

Definition at line 286 of file port.h.

#### 5.2.2.14 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](#)

## 5.3 serdes\_fields\_t Struct Reference

Serdes fields.

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

## Data Fields

- `u32 ob_post0`
- `u32 ob_sr`

### 5.3.1 Detailed Description

Serdes fields.

Definition at line 357 of file `vtss_init_api.h`.

### 5.3.2 Field Documentation

#### 5.3.2.1 `ob_post0`

`u32 serdes_fields_t::ob_post0`

Trace length

Definition at line 358 of file `vtss_init_api.h`.

#### 5.3.2.2 `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`

## 5.4 `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`
- `struct tag_vtss_fdma_list * next`

### 5.4.1 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 346 of file vtss\_fdma\_api.h.

### 5.4.2 Field Documentation

#### 5.4.2.1 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 415 of file vtss\_fdma\_api.h.

#### 5.4.2.2 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 441 of file vtss\_fdma\_api.h.

#### 5.4.2.3 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 454 of file vtss\_fdma\_api.h.

#### 5.4.2.4 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 465 of file vtss\_fdma\_api.h.

#### 5.4.2.5 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 508 of file vtss\_fdma\_api.h.

#### 5.4.2.6 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 515 of file vtss\_fdma\_api.h.

#### 5.4.2.7 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 716 of file vtss\_fdma\_api.h.

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

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

## 5.5 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

### 5.5.1 Detailed Description

Frame data for VTSS\_ACE\_TYPE\_ARP.

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

### 5.5.2 Field Documentation

#### 5.5.2.1 smac

`vtss_ace_u48_t vtss_ace_frame_arp_t::smac`

SMAC

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

#### 5.5.2.2 arp

`vtss_ace_bit_t vtss_ace_frame_arp_t::arp`

Opcode ARP/RARP

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

#### 5.5.2.3 req

`vtss_ace_bit_t vtss_ace_frame_arp_t::req`

Opcode request/reply

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

#### 5.5.2.4 unknown

`vtss_ace_bit_t vtss_ace_frame_arp_t::unknown`

Opcode unknown

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

### 5.5.2.5 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.

### 5.5.2.6 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.

### 5.5.2.7 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.

### 5.5.2.8 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.

### 5.5.2.9 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.

### 5.5.2.10 sip

`vtss_ace_ip_t vtss_ace_frame_arp_t::sip`

Sender IP address

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

### 5.5.2.11 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`

## 5.6 `vtss_ace_frame_etype_t` Struct Reference

Frame data for `VTSS_ACE_TYPE_ETYPE`.

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

### Data Fields

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

### 5.6.1 Detailed Description

Frame data for `VTSS_ACE_TYPE_ETYPE`.

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

### 5.6.2 Field Documentation

### 5.6.2.1 dmac

`vtss_ace_u48_t vtss_ace_frame_etype_t::dmac`

DMAC

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

### 5.6.2.2 smac

`vtss_ace_u48_t vtss_ace_frame_etype_t::smac`

SMAC

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

### 5.6.2.3 etype

`vtss_ace_u16_t vtss_ace_frame_etype_t::etype`

Ethernet Type value

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

### 5.6.2.4 data

`vtss_ace_u16_t vtss_ace_frame_etype_t::data`

MAC data

Definition at line 427 of file vtss\_security\_api.h.

### 5.6.2.5 ptp

`vtss_ace_ptp_t vtss_ace_frame_etype_t::ptp`

PTP header filtering (overrides smac byte 2,4 and data fields)

Definition at line 429 of file vtss\_security\_api.h.

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

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

## 5.7 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](#)
- [vtss\\_ace\\_ptp\\_t ptp](#)
- [vtss\\_ace\\_sip\\_smac\\_t sip\\_smac](#)

### 5.7.1 Detailed Description

Frame data for VTSS\_ACE\_TYPE\_IPV4.

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

### 5.7.2 Field Documentation

#### 5.7.2.1 ttl

```
vtss_ace_bit_t vtss_ace_frame_ipv4_t::ttl
```

TTL zero

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

### 5.7.2.2 fragment

`vtss_ace_bit_t` `vtss_ace_frame_ipv4_t::fragment`

Fragment

Definition at line 469 of file `vtss_security_api.h`.

### 5.7.2.3 options

`vtss_ace_bit_t` `vtss_ace_frame_ipv4_t::options`

Header options

Definition at line 470 of file `vtss_security_api.h`.

### 5.7.2.4 ds

`vtss_ace_u8_t` `vtss_ace_frame_ipv4_t::ds`

DS field

Definition at line 471 of file `vtss_security_api.h`.

### 5.7.2.5 proto

`vtss_ace_u8_t` `vtss_ace_frame_ipv4_t::proto`

Protocol

Definition at line 472 of file `vtss_security_api.h`.

### 5.7.2.6 sip

`vtss_ace_ip_t` `vtss_ace_frame_ipv4_t::sip`

Source IP address

Definition at line 473 of file `vtss_security_api.h`.

### 5.7.2.7 dip

`vtss_ace_ip_t` `vtss_ace_frame_ipv4_t::dip`

Destination IP address

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

### 5.7.2.8 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.

### 5.7.2.9 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.

### 5.7.2.10 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.

### 5.7.2.11 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.

### 5.7.2.12 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.

### 5.7.2.13 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.

### 5.7.2.14 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.

### 5.7.2.15 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.

### 5.7.2.16 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.

### 5.7.2.17 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`.

### 5.7.2.18 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`.

### 5.7.2.19 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`.

### 5.7.2.20 ptp

`vtss_ace_ptp_t` `vtss_ace_frame_ipv4_t::ptp`

PTP filtering (overrides sip field)

Definition at line 488 of file `vtss_security_api.h`.

### 5.7.2.21 sip\_smac

`vtss_ace_sip_smac_t` `vtss_ace_frame_ipv4_t::sip_smac`

SIP/SMAC matching (overrides sip field)

Definition at line 489 of file `vtss_security_api.h`.

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

- `vtss_api/include/vtss_security_api.h`

## 5.8 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](#)
- [vtss\\_ace\\_ptp\\_t ptp](#)

### 5.8.1 Detailed Description

Frame data for VTSS\_ACE\_TYPE\_IPV6.

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

### 5.8.2 Field Documentation

#### 5.8.2.1 proto

```
vtss_ace_u8_t vtss_ace_frame_ipv6_t::proto
```

IPv6 protocol

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

### 5.8.2.2 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`.

### 5.8.2.3 ttl

`vtss_ace_bit_t vtss_ace_frame_ipv6_t::ttl`

TTL zero

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

### 5.8.2.4 ds

`vtss_ace_u8_t vtss_ace_frame_ipv6_t::ds`

DS field

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

### 5.8.2.5 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`.

### 5.8.2.6 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`.

### 5.8.2.7 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`.

### 5.8.2.8 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`.

### 5.8.2.9 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`.

### 5.8.2.10 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`.

### 5.8.2.11 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`.

### 5.8.2.12 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.

### 5.8.2.13 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.

### 5.8.2.14 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.

### 5.8.2.15 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.

### 5.8.2.16 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.

### 5.8.2.17 ptp

`vtss_ace_ptp_t` `vtss_ace_frame_ipv6_t::ptp`

PTP filtering (overrides sip byte 0-3)

Definition at line 513 of file `vtss_security_api.h`.

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

- `vtss_api/include/vtss_security_api.h`

## 5.9 vtss\_ace\_frame\_llc\_t Struct Reference

Frame data for `VTSS_ACE_TYPE_LLCC`.

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

### Data Fields

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

### 5.9.1 Detailed Description

Frame data for `VTSS_ACE_TYPE_LLCC`.

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

### 5.9.2 Field Documentation

#### 5.9.2.1 dmac

`vtss_ace_u48_t` `vtss_ace_frame_llc_t::dmac`

DMAC

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

### 5.9.2.2 smac

`vtss_ace_u48_t vtss_ace_frame_llc_t::smac`

SMAC

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

### 5.9.2.3 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`

## 5.10 `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`

### 5.10.1 Detailed Description

Frame data for VTSS\_ACE\_TYPE\_SNAP.

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

### 5.10.2 Field Documentation

### 5.10.2.1 dmac

`vtss_ace_u48_t vtss_ace_frame_snap_t::dmac`

DMAC

Definition at line 444 of file vtss\_security\_api.h.

### 5.10.2.2 smac

`vtss_ace_u48_t vtss_ace_frame_snap_t::smac`

SMAC

Definition at line 445 of file vtss\_security\_api.h.

### 5.10.2.3 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](#)

## 5.11 vtss\_ace\_ptp\_t Struct Reference

PTP header filtering.

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

### Data Fields

- `BOOL enable`
- `vtss_ace_u32_t header`

### 5.11.1 Detailed Description

PTP header filtering.

Definition at line 391 of file vtss\_security\_api.h.

### 5.11.2 Field Documentation

#### 5.11.2.1 enable

`BOOL vtss_ace_ptp_t::enable`

Enable PTP header filtering

Definition at line 393 of file vtss\_security\_api.h.

#### 5.11.2.2 header

`vtss_ace_u32_t vtss_ace_ptp_t::header`

PTP header byte 0, 1, 4 and 6

Definition at line 394 of file vtss\_security\_api.h.

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

- `vtss_api/include/vtss_security_api.h`

## 5.12 vtss\_ace\_sip\_smac\_t Struct Reference

SIP/SMAC filtering.

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

### Data Fields

- `BOOL enable`
- `vtss_ip_t sip`
- `vtss_mac_t smac`

#### 5.12.1 Detailed Description

SIP/SMAC filtering.

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

## 5.12.2 Field Documentation

### 5.12.2.1 enable

`BOOL vtss_ace_sip_smac_t::enable`

Enable SIP/SMAC filtering

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

### 5.12.2.2 sip

`vtss_ip_t vtss_ace_sip_smac_t::sip`

SIP

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

### 5.12.2.3 smac

`vtss_mac_t vtss_ace_sip_smac_t::smac`

SMAC

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

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

- `vtss_api/include/vtss_security_api.h`

## 5.13 vtss\_ace\_t Struct Reference

Access Control Entry.

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

## Data Fields

- `vtss_ace_id_t id`
- `BOOL port_list [VTSS_PORT_ARRAY_SIZE]`
- `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`

### 5.13.1 Detailed Description

Access Control Entry.

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

### 5.13.2 Field Documentation

#### 5.13.2.1 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.

#### 5.13.2.2 port\_list

`BOOL vtss_ace_t::port_list [VTSS_PORT_ARRAY_SIZE]`

Port list

Definition at line 530 of file vtss\_security\_api.h.

### 5.13.2.3 policy

`vtss_ace_u8_t vtss_ace_t::policy`

Policy number

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

### 5.13.2.4 type

`vtss_ace_type_t vtss_ace_t::type`

ACE frame type

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

### 5.13.2.5 action

`vtss_acl_action_t vtss_ace_t::action`

ACE action

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

### 5.13.2.6 dmac\_mc

`vtss_ace_bit_t vtss_ace_t::dmac_mc`

Multicast DMAC

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

### 5.13.2.7 dmac\_bc

`vtss_ace_bit_t vtss_ace_t::dmac_bc`

Broadcast DMAC

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

### 5.13.2.8 vlan

`vtss_ace_vlan_t` `vtss_ace_t::vlan`

VLAN Tag

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

### 5.13.2.9 etype

`vtss_ace_frame_etype_t` `vtss_ace_t::etype`

VTSS\_ACE\_TYPE\_ETYPE

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

### 5.13.2.10 llc

`vtss_ace_frame_llc_t` `vtss_ace_t::llc`

VTSS\_ACE\_TYPE\_LLCC

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

### 5.13.2.11 snap

`vtss_ace_frame_snap_t` `vtss_ace_t::snap`

VTSS\_ACE\_TYPE\_SNAP

Definition at line 546 of file vtss\_security\_api.h.

### 5.13.2.12 arp

`vtss_ace_frame_arp_t` `vtss_ace_t::arp`

VTSS\_ACE\_TYPE\_ARP

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

## 5.13.2.13 ipv4

```
vtss_ace_frame_ipv4_t vtss_ace_t::ipv4
```

VTSS\_ACE\_TYPE\_IPV4

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

## 5.13.2.14 ipv6

```
vtss_ace_frame_ipv6_t vtss_ace_t::ipv6
```

VTSS\_ACE\_TYPE\_IPV6

Definition at line 549 of file vtss\_security\_api.h.

## 5.13.2.15 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](#)

## 5.14 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](#)
- [vtss\\_ace\\_bit\\_t tagged](#)

### 5.14.1 Detailed Description

ACE VLAN information.

Definition at line 411 of file vtss\_security\_api.h.

---

### 5.14.2 Field Documentation

#### 5.14.2.1 vid

`vtss_ace_vid_t` `vtss_ace_vlan_t::vid`

VLAN ID (12 bit)

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

#### 5.14.2.2 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`.

#### 5.14.2.3 cfi

`vtss_ace_bit_t` `vtss_ace_vlan_t::cfi`

CFI/DEI

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

#### 5.14.2.4 tagged

`vtss_ace_bit_t` `vtss_ace_vlan_t::tagged`

Tagged/untagged frame

Definition at line 417 of file `vtss_security_api.h`.

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

- `vtss_api/include/vtss_security_api.h`

## 5.15 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`
- `vtss_acl_port_action_t port_action`
- `BOOL port_list[VTSS_PORT_ARRAY_SIZE]`
- `BOOL mirror`
- `vtss_acl_ptp_action_t ptp_action`
- `vtss_acl_ptp_action_conf_t ptp`

### 5.15.1 Detailed Description

ACL Action.

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

### 5.15.2 Field Documentation

#### 5.15.2.1 `cpu`

```
BOOL vtss_acl_action_t::cpu
```

Forward to CPU

Definition at line 240 of file vtss\_security\_api.h.

#### 5.15.2.2 `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.

### 5.15.2.3 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.

### 5.15.2.4 police

`BOOL` `vtss_acl_action_t::police`

Enable policer

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

### 5.15.2.5 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.

### 5.15.2.6 learn

`BOOL` `vtss_acl_action_t::learn`

Allow learning

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

### 5.15.2.7 port\_action

`vtss_acl_port_action_t` `vtss_acl_action_t::port_action`

Port action

Definition at line 258 of file vtss\_security\_api.h.

### 5.15.2.8 port\_list

```
BOOL vtss_acl_action_t::port_list[VTSS_PORT_ARRAY_SIZE]
```

Egress port list

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

### 5.15.2.9 mirror

```
BOOL vtss_acl_action_t::mirror
```

Enable mirroring

Definition at line 260 of file vtss\_security\_api.h.

### 5.15.2.10 ptp\_action

```
vtss_acl_ptp_action_t vtss_acl_action_t::ptp_action
```

PTP action

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

### 5.15.2.11 ptp

```
vtss_acl_ptp_action_conf_t vtss_acl_action_t::ptp
```

PTP configuration

Definition at line 264 of file vtss\_security\_api.h.

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

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

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

ACL policer configuration.

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

## Data Fields

- `BOOL bit_rate_enable`
- `vtss_bitrate_t bit_rate`
- `vtss_packet_rate_t rate`

### 5.16.1 Detailed Description

ACL policer configuration.

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

### 5.16.2 Field Documentation

#### 5.16.2.1 `bit_rate_enable`

`BOOL vtss_acl_policer_conf_t::bit_rate_enable`

Use bit rate policing instead of packet rate

Definition at line 157 of file `vtss_security_api.h`.

#### 5.16.2.2 `bit_rate`

`vtss_bitrate_t vtss_acl_policer_conf_t::bit_rate`

Bit rate

Definition at line 158 of file `vtss_security_api.h`.

#### 5.16.2.3 `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`

## 5.17 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](#)

#### 5.17.1 Detailed Description

ACL port configuration.

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

#### 5.17.2 Field Documentation

##### 5.17.2.1 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.

##### 5.17.2.2 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](#)

## 5.18 vtss\_acl\_ptp\_action\_conf\_t Struct Reference

ACL PTP action configuration.

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

## Data Fields

- `vtss_acl_ptp_rsp_t response`
- `i8 log_message_interval`
- `BOOL copy_smac_to_dmac`
- `BOOL set_smac_to_port_mac`
- `u8 dom_sel`

### 5.18.1 Detailed Description

ACL PTP action configuration.

Definition at line 228 of file `vtss_security_api.h`.

### 5.18.2 Field Documentation

#### 5.18.2.1 `response`

`vtss_acl_ptp_rsp_t vtss_acl_ptp_action_conf_t::response`

PTP Delay\_Req/Response action

Definition at line 229 of file `vtss_security_api.h`.

#### 5.18.2.2 `log_message_interval`

`i8 vtss_acl_ptp_action_conf_t::log_message_interval`

PTP logMessageInterval [-8,7] returned in the Delay\_Resp message

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

#### 5.18.2.3 `copy_smac_to_dmac`

`BOOL vtss_acl_ptp_action_conf_t::copy_smac_to_dmac`

PTP DMAC operation

Definition at line 231 of file `vtss_security_api.h`.

#### 5.18.2.4 set\_smac\_to\_port\_mac

`BOOL vtss_acl_ptp_action_conf_t::set_smac_to_port_mac`

PTP SMAC operation

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

#### 5.18.2.5 dom\_sel

`u8 vtss_acl_ptp_action_conf_t::dom_sel`

PTP domain selector. PTP\_DOM\_SEL indexes the PTP configuration

Definition at line 233 of file vtss\_security\_api.h.

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

- `vtss_api/include/vtss_security_api.h`

## 5.19 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`

#### 5.19.1 Detailed Description

Aggregation traffic distribution mode.

Definition at line 39 of file l2\_types.h.

#### 5.19.2 Field Documentation

### 5.19.2.1 smac\_enable

`BOOL vtss_aggr_mode_t::smac_enable`

Source MAC address

Definition at line 41 of file `I2_types.h`.

### 5.19.2.2 dmac\_enable

`BOOL vtss_aggr_mode_t::dmac_enable`

Destination MAC address

Definition at line 42 of file `I2_types.h`.

### 5.19.2.3 sip\_dip\_enable

`BOOL vtss_aggr_mode_t::sip_dip_enable`

Source and destination IP address

Definition at line 43 of file `I2_types.h`.

### 5.19.2.4 sport\_dport\_enable

`BOOL vtss_aggr_mode_t::sport_dport_enable`

Source and destination UDP/TCP port

Definition at line 44 of file `I2_types.h`.

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

- `vtss_api/include/vtss/api/I2_types.h`

## 5.20 vtss\_aneg\_t Struct Reference

Auto negotiation struct.

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

## Data Fields

- `BOOL obey_pause`
- `BOOL generate_pause`

### 5.20.1 Detailed Description

Auto negotiation struct.

Definition at line 482 of file types.h.

### 5.20.2 Field Documentation

#### 5.20.2.1 `obey_pause`

`BOOL vtss_aneg_t::obey_pause`

This port should obey PAUSE frames

Definition at line 484 of file types.h.

#### 5.20.2.2 `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`

## 5.21 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`

### 5.21.1 Detailed Description

API lock structure.

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

### 5.21.2 Field Documentation

#### 5.21.2.1 inst

`vtss_inst_t vtss_api_lock_t::inst`

Target instance reference

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

#### 5.21.2.2 function

`const char* vtss_api_lock_t::function`

Function name

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

#### 5.21.2.3 file

`const char* vtss_api_lock_t::file`

File name

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

## 5.21.2.4 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](#)

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

Basic counters structure.

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

### Data Fields

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

#### 5.22.1 Detailed Description

Basic counters structure.

Definition at line 377 of file vtss\_port\_api.h.

#### 5.22.2 Field Documentation

##### 5.22.2.1 rx\_frames

```
u32 vtss_basic_counters_t::rx_frames
```

Rx frames

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

### 5.22.2.2 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](#)

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

Chip ID.

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

### Data Fields

- [u16 part\\_number](#)
- [u16 revision](#)

### 5.23.1 Detailed Description

Chip ID.

Definition at line 401 of file vtss\_misc\_api.h.

### 5.23.2 Field Documentation

#### 5.23.2.1 part\_number

```
u16 vtss_chip_id_t::part_number
```

BCD encoded part number

Definition at line 403 of file vtss\_misc\_api.h.

### 5.23.2.2 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](#)

## 5.24 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

### 5.24.1 Detailed Description

Debug information structure.

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

### 5.24.2 Field Documentation

#### 5.24.2.1 layer

```
vtss_debug_layer_t vtss_debug_info_t::layer
```

Layer

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

### 5.24.2.2 group

```
vtss_debug_group_t vtss_debug_info_t::group
```

Function group

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

### 5.24.2.3 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.

### 5.24.2.4 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.

### 5.24.2.5 full

```
BOOL vtss_debug_info_t::full
```

Full information dump

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

### 5.24.2.6 clear

```
BOOL vtss_debug_info_t::clear
```

Clear counters

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

### 5.24.2.7 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`

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

API debug lock structure.

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

### Data Fields

- `vtss_chip_no_t chip_no`

### 5.25.1 Detailed Description

API debug lock structure.

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

### 5.25.2 Field Documentation

#### 5.25.2.1 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`

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

Destination group port configuration.

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

### Data Fields

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

#### 5.26.1 Detailed Description

Destination group port configuration.

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

#### 5.26.2 Field Documentation

##### 5.26.2.1 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](#)

## 5.27 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](#)

### 5.27.1 Detailed Description

Dual leaky buckets policer configuration.

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

### 5.27.2 Field Documentation

#### 5.27.2.1 type

`vtss_policer_type_t vtss_dlb_policer_conf_t::type`

Policer type

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

#### 5.27.2.2 enable

`BOOL vtss_dlb_policer_conf_t::enable`

Enable/disable policer

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

#### 5.27.2.3 cm

`BOOL vtss_dlb_policer_conf_t::cm`

Colour Mode (TRUE means colour aware)

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

#### 5.27.2.4 cf

`BOOL vtss_dlb_policer_conf_t::cf`

Coupling Flag

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

### 5.27.2.5 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`.

### 5.27.2.6 cir

`vtss_bitrate_t vtss_dlb_policer_conf_t::cir`

Committed Information Rate

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

### 5.27.2.7 cbs

`vtss_burst_level_t vtss_dlb_policer_conf_t::cbs`

Committed Burst Size

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

### 5.27.2.8 eir

`vtss_bitrate_t vtss_dlb_policer_conf_t::eir`

Excess Information Rate

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

### 5.27.2.9 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`

## 5.28 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`

#### 5.28.1 Detailed Description

EEE port configuration.

Definition at line 1219 of file vtss\_misc\_api.h.

#### 5.28.2 Field Documentation

##### 5.28.2.1 eee\_ena

```
BOOL vtss_eee_port_conf_t::eee_ena
```

Enable EEE

Definition at line 1221 of file vtss\_misc\_api.h.

##### 5.28.2.2 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.

### 5.28.2.3 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.

### 5.28.2.4 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.

### 5.28.2.5 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](#)

## 5.29 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](#)

### 5.29.1 Detailed Description

EEE port counters (JR only)

Definition at line 1246 of file vtss\_misc\_api.h.

### 5.29.2 Field Documentation

#### 5.29.2.1 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.

#### 5.29.2.2 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.

#### 5.29.2.3 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.

#### 5.29.2.4 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.

### 5.29.2.5 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](#)

## 5.30 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](#)

### 5.30.1 Detailed Description

EEE port state (JR only)

Definition at line 1239 of file [vtss\\_misc\\_api.h](#).

### 5.30.2 Field Documentation

#### 5.30.2.1 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](#).

### 5.30.2.2 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](#)

## 5.31 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

#### 5.31.1 Detailed Description

Port protection configuration.

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

#### 5.31.2 Field Documentation

##### 5.31.2.1 type

```
vtss\_eps\_port\_type\_t vtss_eps_port_conf_t::type
```

Protection type

Definition at line 2143 of file vtss\_l2\_api.h.

### 5.31.2.2 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`

## 5.32 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`

### 5.32.1 Detailed Description

eWIS AIS-L consequent actions

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

### 5.32.2 Field Documentation

#### 5.32.2.1 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`.

### 5.32.2.2 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](#)

## 5.33 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`

### 5.33.1 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.

### 5.33.2 Field Documentation

### 5.33.2.1 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.

### 5.33.2.2 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.

### 5.33.2.3 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.

### 5.33.2.4 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.

### 5.33.2.5 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.

### 5.33.2.6 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.

### 5.33.2.7 path\_txti

`vtss_ewis_tti_t` `vtss_ewis_conf_s::path_txti`

Path Trail Trace Identifier Configuration

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

### 5.33.2.8 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.

### 5.33.2.9 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.

### 5.33.2.10 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.

### 5.33.2.11 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`.

### 5.33.2.12 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`.

### 5.33.2.13 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`

## 5.34 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`

### 5.34.1 Detailed Description

eWIS consequent actions

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

### 5.34.2 Field Documentation

#### 5.34.2.1 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`.

#### 5.34.2.2 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`.

#### 5.34.2.3 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`

## 5.35 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`

### 5.35.1 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.

### 5.35.2 Field Documentation

#### 5.35.2.1 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.

#### 5.35.2.2 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.

#### 5.35.2.3 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.

#### 5.35.2.4 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.

### 5.35.2.5 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](#)

## 5.36 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`

### 5.36.1 Detailed Description

eWIS performance counter thresholds.

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

### 5.36.2 Field Documentation

#### 5.36.2.1 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.

### 5.36.2.2 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`.

### 5.36.2.3 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`.

### 5.36.2.4 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`.

### 5.36.2.5 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`

## 5.37 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`

### 5.37.1 Detailed Description

eWIS defects

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

### 5.37.2 Field Documentation

#### 5.37.2.1 dlos\_s

`BOOL vtss_ewis_defects_s::dlos_s`

Loss of signal

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

#### 5.37.2.2 doof\_s

`BOOL vtss_ewis_defects_s::doof_s`

Out of frame

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

### 5.37.2.3 dlof\_s

`BOOL vtss_ewis_defects_s::dlof_s`

Loss of frame

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

### 5.37.2.4 dais\_l

`BOOL vtss_ewis_defects_s::dais_l`

Line alarm indication signal

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

### 5.37.2.5 drdi\_l

`BOOL vtss_ewis_defects_s::drdi_l`

Line remote defect indication

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

### 5.37.2.6 dais\_p

`BOOL vtss_ewis_defects_s::dais_p`

Path alarm indication signal

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

### 5.37.2.7 dlop\_p

`BOOL vtss_ewis_defects_s::dlop_p`

Loss of pointer

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

### 5.37.2.8 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.

### 5.37.2.9 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.

### 5.37.2.10 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.

### 5.37.2.11 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.

### 5.37.2.12 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.

### 5.37.2.13 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`.

### 5.37.2.14 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`

## 5.38 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`

### 5.38.1 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`.

## 5.38.2 Field Documentation

### 5.38.2.1 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.

### 5.38.2.2 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.

### 5.38.2.3 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.

### 5.38.2.4 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.

### 5.38.2.5 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.

### 5.38.2.6 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.

### 5.38.2.7 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.

### 5.38.2.8 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.

### 5.38.2.9 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.

### 5.38.2.10 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.

### 5.38.2.11 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`

## 5.39 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`

### 5.39.1 Detailed Description

eWIS force modes

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

### 5.39.2 Field Documentation

#### 5.39.2.1 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.

### 5.39.2.2 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`.

### 5.39.2.3 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`

## 5.40 `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`

### 5.40.1 Detailed Description

eWIS line force mode

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

### 5.40.2 Field Documentation

#### 5.40.2.1 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.

#### 5.40.2.2 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](#)

## 5.41 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`

#### 5.41.1 Detailed Description

eWIS line TX force mode

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

#### 5.41.2 Field Documentation

### 5.41.2.1 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`.

### 5.41.2.2 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`

## 5.42 `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`

### 5.42.1 Detailed Description

eWIS path force modes

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

### 5.42.2 Field Documentation

#### 5.42.2.1 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.

#### 5.42.2.2 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](#)

## 5.43 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](#)

#### 5.43.1 Detailed Description

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

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

#### 5.43.2 Field Documentation

#### 5.43.2.1 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`.

#### 5.43.2.2 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`.

#### 5.43.2.3 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`.

#### 5.43.2.4 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`.

#### 5.43.2.5 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`

## 5.44 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

#### 5.44.1 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.

#### 5.44.2 Field Documentation

##### 5.44.2.1 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.

##### 5.44.2.2 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.

#### 5.44.2.3 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`.

#### 5.44.2.4 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`.

#### 5.44.2.5 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`

## 5.45 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`

### 5.45.1 Detailed Description

eWIS RDI-L consequent actions

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

### 5.45.2 Field Documentation

#### 5.45.2.1 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.

#### 5.45.2.2 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.

#### 5.45.2.3 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.

#### 5.45.2.4 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](#)

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

signal label configuration

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

### Data Fields

- `u8 exsl`

#### 5.46.1 Detailed Description

signal label configuration

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

#### 5.46.2 Field Documentation

##### 5.46.2.1 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](#)

## 5.47 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`

### 5.47.1 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.

### 5.47.2 Field Documentation

#### 5.47.2.1 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.

#### 5.47.2.2 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.

#### 5.47.2.3 `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`.

#### 5.47.2.4 `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`.

#### 5.47.2.5 `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`.

#### 5.47.2.6 `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`.

#### 5.47.2.7 `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`.

**5.47.2.8 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.

**5.47.2.9 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.

**5.47.2.10 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.

**5.47.2.11 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.

**5.47.2.12 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.

#### 5.47.2.13 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`.

#### 5.47.2.14 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`.

#### 5.47.2.15 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`

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

eWIS status

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

### Data Fields

- `BOOL fault`
- `BOOL link_stat`

#### 5.48.1 Detailed Description

eWIS status

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

## 5.48.2 Field Documentation

### 5.48.2.1 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.

### 5.48.2.2 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](#)

## 5.49 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`

### 5.49.1 Detailed Description

eWIS test configuration

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

## 5.49.2 Field Documentation

### 5.49.2.1 loopback

`BOOL vtss_ewis_test_conf_s::loopback`

loop output from Tx to Rx

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

### 5.49.2.2 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`.

### 5.49.2.3 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`

## 5.50 `vtss_ewis_test_status_s` Struct Reference

eWIS test status

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

### Data Fields

- `u16 tstpat_cnt`
- `BOOL ana_sync`

### 5.50.1 Detailed Description

eWIS test status

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

### 5.50.2 Field Documentation

#### 5.50.2.1 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.

#### 5.50.2.2 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](#)

## 5.51 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](#)

### 5.51.1 Detailed Description

Trail Trace Identifier type.

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

### 5.51.2 Field Documentation

#### 5.51.2.1 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.

#### 5.51.2.2 tti

`u8 vtss_ewis_tti_s::tti[64]`

trace identifier value

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

#### 5.51.2.3 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](#)

## 5.52 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`

### 5.52.1 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.

### 5.52.2 Field Documentation

#### 5.52.2.1 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.

#### 5.52.2.2 tx\_e1

```
u8 vtss_ewis_tx_oh_s::tx_e1
```

Orderwire

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

### 5.52.2.3 tx\_f1

`u8 vtss_ewis_tx_oh_s::tx_f1`

Section User Channel

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

### 5.52.2.4 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.

### 5.52.2.5 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.

### 5.52.2.6 tx\_e2

`u8 vtss_ewis_tx_oh_s::tx_e2`

Orderwire

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

### 5.52.2.7 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.

### 5.52.2.8 tx\_s1

`u8 vtss_ewis_tx_oh_s::tx_s1`

Synchronization messaging

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

### 5.52.2.9 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.

### 5.52.2.10 tx\_c2

`u8 vtss_ewis_tx_oh_s::tx_c2`

Transmitted C2 path label

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

### 5.52.2.11 tx\_f2

`u8 vtss_ewis_tx_oh_s::tx_f2`

Path User Channel

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

### 5.52.2.12 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.

### 5.52.2.13 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](#)

## 5.53 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](#)

### 5.53.1 Detailed Description

eWIS overhead passthru configuration.

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

### 5.53.2 Field Documentation

### 5.53.2.1 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.

### 5.53.2.2 tx\_z0

`BOOL vtss_ewis_tx_passthru_s::tx_z0`

z0 Section growth passthrough

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

### 5.53.2.3 tx\_b1

`BOOL vtss_ewis_tx_passthru_s::tx_b1`

b1 BIP passthrough

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

### 5.53.2.4 tx\_e1

`BOOL vtss_ewis_tx_passthru_s::tx_e1`

e1 order wire passthrough

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

### 5.53.2.5 tx\_f1

`BOOL vtss_ewis_tx_passthru_s::tx_f1`

f1 Section user channel

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

### 5.53.2.6 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.

### 5.53.2.7 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.

### 5.53.2.8 tx\_b2

`BOOL vtss_ewis_tx_passthru_s::tx_b2`

b2 BIP passthrough

Definition at line 256 of file vtss\_wis\_api.h.

### 5.53.2.9 tx\_k1

`BOOL vtss_ewis_tx_passthru_s::tx_k1`

k1 passthrough

Definition at line 257 of file vtss\_wis\_api.h.

### 5.53.2.10 tx\_k2

`BOOL vtss_ewis_tx_passthru_s::tx_k2`

k2 passthrough

Definition at line 258 of file vtss\_wis\_api.h.

### 5.53.2.11 tx\_reil

`BOOL vtss_ewis_tx_passthru_s::tx_reil`

reil passthrough

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

### 5.53.2.12 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.

### 5.53.2.13 tx\_s1

`BOOL vtss_ewis_tx_passthru_s::tx_s1`

Synchronization messaging passthrough

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

### 5.53.2.14 tx\_e2

`BOOL vtss_ewis_tx_passthru_s::tx_e2`

order wire passthrough

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

### 5.53.2.15 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.

### 5.53.2.16 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`

## 5.54 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`

### 5.54.1 Detailed Description

Fan specifications.

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

### 5.54.2 Field Documentation

#### 5.54.2.1 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`.

#### 5.54.2.2 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.

#### 5.54.2.3 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.

#### 5.54.2.4 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.

#### 5.54.2.5 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

## 5.55 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)`

### 5.55.1 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 1820 of file vtss\_fdma\_api.h.

### 5.55.2 Field Documentation

#### 5.55.2.1 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 1830 of file vtss\_fdma\_api.h.

#### 5.55.2.2 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 1843 of file vtss\_fdma\_api.h.

### 5.55.2.3 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 1852 of file vtss\_fdma\_api.h.

### 5.55.2.4 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 1925 of file vtss\_fdma\_api.h.

#### 5.55.2.5 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 1938 of file vtss\_fdma\_api.h.

#### 5.55.2.6 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 1945 of file vtss\_fdma\_api.h.

#### 5.55.2.7 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 1952 of file vtss\_fdma\_api.h.

### 5.55.2.8 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 1964 of file vtss\_fdma\_api.h.

### 5.55.2.9 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 1987 of file vtss\_fdma\_api.h.

### 5.55.2.10 tx\_buf\_cnt

```
u32 vtss_fdma_cfg_t::tx_buf_cnt
```

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

Definition at line 1993 of file vtss\_fdma\_api.h.

### 5.55.2.11 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. 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 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 2015 of file vtss\_fdma\_api.h.

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

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

## 5.56 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 \*cntxt, [vtss\\_fdma\\_list\\_t](#) \*list, [vtss\\_packet\\_rx\\_queue\\_t](#) qu)
- int prio
- [vtss\\_chip\\_no\\_t](#) chip\_no

### 5.56.1 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 1595 of file vtss\_fdma\_api.h.

### 5.56.2 Field Documentation

#### 5.56.2.1 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 1603 of file vtss\_fdma\_api.h.

#### 5.56.2.2 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-ported, only. Luton26: N Serval : N Jaguar2: N Serval2: N ServalT: N

Definition at line 1623 of file vtss\_fdma\_api.h.

### 5.56.2.3 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 1653 of file `vtss_fdma_api.h`.

### 5.56.2.4 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 1676 of file `vtss_fdma_api.h`.

## 5.56.2.5 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-NUL 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 1709 of file vtss\_fdma\_api.h.

## 5.56.2.6 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 1724 of file vtss\_fdma\_api.h.

### 5.56.2.7 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 1745 of file vtss\_fdma\_api.h.

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

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

## 5.57 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]

### 5.57.1 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 2204 of file vtss\_fdma\_api.h.

## 5.57.2 Field Documentation

### 5.57.2.1 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 2224 of file `vtss_fdma_api.h`.

### 5.57.2.2 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 2235 of file `vtss_fdma_api.h`.

### 5.57.2.3 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 2247 of file `vtss_fdma_api.h`.

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

- `vtss_api/include/vtss_fdma_api.h`

## 5.58 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)

### 5.58.1 Detailed Description

FDMA Injection Properties.

Definition at line 742 of file vtss\_fdma\_api.h.

### 5.58.2 Field Documentation

#### 5.58.2.1 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 746 of file vtss\_fdma\_api.h.

#### 5.58.2.2 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 751 of file vtss\_fdma\_api.h.

#### 5.58.2.3 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 779 of file vtss\_fdma\_api.h.

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

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

## 5.59 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`

#### 5.59.1 Detailed Description

GPIO configured mode.

GPIO input modes

Definition at line 2480 of file vtss\_phy\_10g\_api.h.

#### 5.59.2 Field Documentation

##### 5.59.2.1 mode

```
vtss_10g_phy_gpio_t vtss_gpio_10g_gpio_mode_t::mode
```

Mode of this GPIO pin

Definition at line 2482 of file vtss\_phy\_10g\_api.h.

##### 5.59.2.2 port

```
vtss_port_no_t vtss_gpio_10g_gpio_mode_t::port
```

In case of VTSS\_10G\_PHY\_GPIO\_WIS\_INT mode, this is the interrupt port number that is related to this GPIO In case of VTSS\_10G\_PHY\_GPIO\_PCS\_RX\_FAULT mode, this is the PCS status port number that is related to this GPIO

Definition at line 2483 of file vtss\_phy\_10g\_api.h.

### 5.59.2.3 input

`vtss_gpio_10g_input_t` `vtss_gpio_10g_gpio_mode_t::input`

GPIO input modes

Definition at line 2485 of file vtss\_phy\_10g\_api.h.

### 5.59.2.4 in\_sig

`vtss_gpio_10g_gpio_intr_sgnl_t` `vtss_gpio_10g_gpio_mode_t::in_sig`

Internal signal that to be routed through GPIO

Definition at line 2486 of file vtss\_phy\_10g\_api.h.

### 5.59.2.5 p\_gpio

`vtss_gpio_10g_no_t` `vtss_gpio_10g_gpio_mode_t::p_gpio`

Per channel GPIO number

Definition at line 2487 of file vtss\_phy\_10g\_api.h.

### 5.59.2.6 c\_intrpt

`vtss_gpio_10g_chan_intrpt_t` `vtss_gpio_10g_gpio_mode_t::c_intrpt`

Per Channel interrupt,

Definition at line 2488 of file vtss\_phy\_10g\_api.h.

### 5.59.2.7 source

`u16` `vtss_gpio_10g_gpio_mode_t::source`

source of GPIO, approriate value from GPIO\_OUT\_CFG\_X register field GPIO\_X\_SEL

Definition at line 2489 of file vtss\_phy\_10g\_api.h.

#### 5.59.2.8 aggr\_intrpt

```
u32 vtss_gpio_10g_gpio_mode_t::aggr_intrpt
```

Bitmask corresponds to aggregated interrupt

Definition at line 2490 of file vtss\_phy\_10g\_api.h.

#### 5.59.2.9 use\_as\_intrpt

```
BOOL vtss_gpio_10g_gpio_mode_t::use_as_intrpt
```

Change in GPIO generates GPIO interrupt ,supported on MALIBU

Definition at line 2491 of file vtss\_phy\_10g\_api.h.

#### 5.59.2.10 p\_gpio\_intrpt

```
BOOL vtss_gpio_10g_gpio_mode_t::p_gpio_intrpt
```

Port GPIO Change interrupt

Definition at line 2492 of file vtss\_phy\_10g\_api.h.

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

- vtss\_api/include/vtss\_phy\_10g\_api.h

## 5.60 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](#)

### 5.60.1 Detailed Description

Initialization configuration.

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

### 5.60.2 Field Documentation

#### 5.60.2.1 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.

#### 5.60.2.2 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.

#### 5.60.2.3 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.

#### 5.60.2.4 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.

### 5.60.2.5 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.

### 5.60.2.6 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.

### 5.60.2.7 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.

### 5.60.2.8 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.

### 5.60.2.9 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.

### 5.60.2.10 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.

### 5.60.2.11 warm\_start\_enable

`BOOL vtss_init_conf_t::warm_start_enable`

Allow warm start

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

### 5.60.2.12 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.

### 5.60.2.13 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.

### 5.60.2.14 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.

### 5.60.2.15 pi

`vtss_pi_conf_t` `vtss_init_conf_t::pi`

Parallel Interface configuration

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

### 5.60.2.16 serdes

`vtss_serdes_macro_conf_t` `vtss_init_conf_t::serdes`

Serdes macro configuration

Definition at line 550 of file `vtss_init_api.h`.

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

- `vtss_api/include/vtss_init_api.h`

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

Create structure.

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

### Data Fields

- `vtss_target_type_t target`

### 5.61.1 Detailed Description

Create structure.

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

### 5.61.2 Field Documentation

### 5.61.2.1 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](#)

## 5.62 vtss\_intr\_t Struct Reference

Interrupt source structure.

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

### Data Fields

- `BOOL link_change [VTSS_PORT_ARRAY_SIZE]`

### 5.62.1 Detailed Description

Interrupt source structure.

Definition at line 913 of file vtss\_misc\_api.h.

### 5.62.2 Field Documentation

#### 5.62.2.1 link\_change

```
BOOL vtss_intr_t::link_change[VTSS_PORT_ARRAY_SIZE]
```

Applies to XAUI, 100FX and 1000X ports

Definition at line 914 of file vtss\_misc\_api.h.

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

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

## 5.63 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`

### 5.63.1 Detailed Description

Either an IPv4 or IPv6 address.

Definition at line 813 of file types.h.

### 5.63.2 Field Documentation

#### 5.63.2.1 type

```
vtss_ip_type_t vtss_ip_addr_t::type
```

Union type

Definition at line 814 of file types.h.

#### 5.63.2.2 ipv4

```
vtss_ipv4_t vtss_ip_addr_t::ipv4
```

IPv4 address

Definition at line 816 of file types.h.

### 5.63.2.3 ipv6

`vtss_ipv6_t vtss_ip_addr_t::ipv6`

IPv6 address

Definition at line 817 of file types.h.

### 5.63.2.4 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](#)

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

IPv6 network.

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

### Data Fields

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

### 5.64.1 Detailed Description

IPv6 network.

Definition at line 836 of file types.h.

### 5.64.2 Field Documentation

#### 5.64.2.1 address

`vtss_ip_addr_t vtss_ip_network_t::address`

Network address

Definition at line 838 of file types.h.

### 5.64.2.2 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`

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

IPv4 network.

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

### Data Fields

- `vtss_ipv4_t address`
- `vtss_prefix_size_t prefix_size`

#### 5.65.1 Detailed Description

IPv4 network.

Definition at line 822 of file types.h.

#### 5.65.2 Field Documentation

##### 5.65.2.1 address

`vtss_ipv4_t vtss_ipv4_network_t::address`

Network address

Definition at line 824 of file types.h.

### 5.65.2.2 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`

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

IPv4 unicast routing entry.

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

### Data Fields

- `vtss_ipv4_network_t network`
- `vtss_ipv4_t destination`

### 5.66.1 Detailed Description

IPv4 unicast routing entry.

Definition at line 852 of file types.h.

### 5.66.2 Field Documentation

#### 5.66.2.1 network

`vtss_ipv4_network_t vtss_ipv4_uc_t::network`

Network to route

Definition at line 854 of file types.h.

### 5.66.2.2 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`

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

IPv6 network.

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

### Data Fields

- `vtss_ipv6_t address`
- `vtss_prefix_size_t prefix_size`

### 5.67.1 Detailed Description

IPv6 network.

Definition at line 829 of file types.h.

### 5.67.2 Field Documentation

#### 5.67.2.1 address

`vtss_ipv6_t vtss_ipv6_network_t::address`

Network address

Definition at line 831 of file types.h.

### 5.67.2.2 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](#)

## 5.68 vtss\_ipv6\_t Struct Reference

IPv6 address/mask.

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

### Data Fields

- [u8 addr \[16\]](#)

### 5.68.1 Detailed Description

IPv6 address/mask.

Definition at line 797 of file types.h.

### 5.68.2 Field Documentation

#### 5.68.2.1 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](#)

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

IPv6 routing entry.

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

### Data Fields

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

#### 5.69.1 Detailed Description

IPv6 routing entry.

Definition at line 860 of file types.h.

#### 5.69.2 Field Documentation

##### 5.69.2.1 network

```
vtss_ipv6_network_t vtss_ipv6_uc_t::network
```

Network to route

Definition at line 862 of file types.h.

##### 5.69.2.2 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](#)

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

Interrupt configuration options.

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

## Data Fields

- `BOOL external`
- `u8 destination`

### 5.70.1 Detailed Description

Interrupt configuration options.

Definition at line 972 of file `vtss_misc_api.h`.

### 5.70.2 Field Documentation

#### 5.70.2.1 `external`

`BOOL vtss_irq_conf_t::external`

Redirect to external IRQ

Definition at line 973 of file `vtss_misc_api.h`.

#### 5.70.2.2 `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`

## 5.71 `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`

### 5.71.1 Detailed Description

Interrupt status structure.

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

### 5.71.2 Field Documentation

#### 5.71.2.1 `active`

`u32 vtss_irq_status_t::active`

Bitmap for pending IRQs (VTSS\_IRQ\_xxx)

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

#### 5.71.2.2 `raw_ident`

`u32 vtss_irq_status_t::raw_ident`

RAW (target dependent) bitmap for active pending IRQs

Definition at line 982 of file vtss\_misc\_api.h.

#### 5.71.2.3 `raw_status`

`u32 vtss_irq_status_t::raw_status`

RAW (target dependent) bitmap for all pending IRQs

Definition at line 983 of file vtss\_misc\_api.h.

### 5.71.2.4 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](#)

## 5.72 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

### 5.72.1 Detailed Description

Routing interface statics counter.

Definition at line 886 of file types.h.

### 5.72.2 Field Documentation

#### 5.72.2.1 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.

### 5.72.2.2 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.

### 5.72.2.3 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.

### 5.72.2.4 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.

### 5.72.2.5 ipv4uc\_transmitted\_octets

`u64 vtss_l3_counters_t::ipv4uc_transmitted_octets`

IPv4UC octets transmitted

Definition at line 892 of file types.h.

### 5.72.2.6 ipv4uc\_transmitted\_frames

`u64 vtss_l3_counters_t::ipv4uc_transmitted_frames`

IPv4UC frames transmitted

Definition at line 893 of file types.h.

### 5.72.2.7 ipv6uc\_transmitted\_octets

`u64 vtss_l3_counters_t::ipv6uc_transmitted_octets`

IPv6UC octets transmitted

Definition at line 894 of file types.h.

### 5.72.2.8 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`

## 5.73 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`

### 5.73.1 Detailed Description

Structure for Get PHY LC-PLL status.

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

### 5.73.2 Field Documentation

### 5.73.2.1 lock\_status

`u8 vtss_lcppll_status_t::lock_status`

PLL lock status

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

### 5.73.2.2 cal\_done

`u8 vtss_lcppll_status_t::cal_done`

Calibration status

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

### 5.73.2.3 cal\_error

`u8 vtss_lcppll_status_t::cal_error`

Calibration Error indication

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

### 5.73.2.4 fsm\_lock

`u8 vtss_lcppll_status_t::fsm_lock`

FSM lock status

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

### 5.73.2.5 fsm\_stat

`u8 vtss_lcppll_status_t::fsm_stat`

FSM internal status

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

### 5.73.2.6 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`

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

Learning mode.

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

### Data Fields

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

#### 5.74.1 Detailed Description

Learning mode.

Definition at line 379 of file `vtss_l2_api.h`.

#### 5.74.2 Field Documentation

##### 5.74.2.1 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`.

### 5.74.2.2 cpu

`BOOL vtss_learn_mode_t::cpu`

Learn frames copied to CPU (default disabled)

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

### 5.74.2.3 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`

## 5.75 vtss\_mac\_t Struct Reference

MAC Address.

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

### Data Fields

- `u8 addr [6]`

### 5.75.1 Detailed Description

MAC Address.

Definition at line 643 of file types.h.

### 5.75.2 Field Documentation

### 5.75.2.1 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](#)

## 5.76 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](#)

### 5.76.1 Detailed Description

MAC address entry.

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

### 5.76.2 Field Documentation

### 5.76.2.1 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.

### 5.76.2.2 destination

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

Dest. ports

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

### 5.76.2.3 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.

### 5.76.2.4 locked

`BOOL` `vtss_mac_table_entry_t::locked`

Locked/static flag

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

### 5.76.2.5 aged

`BOOL` `vtss_mac_table_entry_t::aged`

Age flag

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

### 5.76.2.6 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.

### 5.76.2.7 enable

`BOOL` `vtss_mac_table_entry_t::enable`

Use (UPSID, UPSPN) when adding entry

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

### 5.76.2.8 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.

### 5.76.2.9 upsid

`vtss_vstax_upsid_t` `vtss_mac_table_entry_t::upsid`

UPS identifier

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

### 5.76.2.10 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.

### 5.76.2.11 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](#)

## 5.77 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](#)

### 5.77.1 Detailed Description

MAC address table status.

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

### 5.77.2 Field Documentation

#### 5.77.2.1 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](#).

### 5.77.2.2 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`.

### 5.77.2.3 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`.

### 5.77.2.4 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`

## 5.78 `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`

### 5.78.1 Detailed Description

Mirror configuration.

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

### 5.78.2 Field Documentation

#### 5.78.2.1 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.

#### 5.78.2.2 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.

#### 5.78.2.3 tag

`vtss_mirror_tag_t` `vtss_mirror_conf_t::tag`

Mirror tag type

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

#### 5.78.2.4 vid

`vtss_vid_t` `vtss_mirror_conf_t::vid`

Mirror tag VID

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

### 5.78.2.5 pcp

`vtss_tagprior_t vtss_mirror_conf_t::pcp`

Mirror tag PCP

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

### 5.78.2.6 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`

## 5.79 `vtss_mtimer_t` Struct Reference

Timer structure.

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

### Data Fields

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

### 5.79.1 Detailed Description

Timer structure.

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

### 5.79.2 Field Documentation

### 5.79.2.1 timeout

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

Timeout

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

### 5.79.2.2 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](#)

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

NPI configuration.

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

### Data Fields

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

### 5.80.1 Detailed Description

NPI configuration.

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

### 5.80.2 Field Documentation

### 5.80.2.1 enable

`BOOL vtss_npi_conf_t::enable`

Enable NPI port

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

### 5.80.2.2 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`

## 5.81 `vtss_os_timestamp_t` Struct Reference

`#include <vtss_misc_api.h>`

### Data Fields

- `unsigned int hw_cnt`

### 5.81.1 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`.

### 5.81.2 Field Documentation

### 5.81.2.1 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](#)

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

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

### Data Fields

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

### 5.82.1 Detailed Description

The packet DMA configuration.

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

### 5.82.2 Field Documentation

#### 5.82.2.1 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](#)

## 5.83 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`

#### 5.83.1 Detailed Description

Information about frame.

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

#### 5.83.2 Field Documentation

##### 5.83.2.1 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.

##### 5.83.2.2 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.

### 5.83.2.3 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.

### 5.83.2.4 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.

### 5.83.2.5 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.

### 5.83.2.6 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](#)

## 5.84 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`

### 5.84.1 Detailed Description

Packet information for each port.

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

### 5.84.2 Field Documentation

#### 5.84.2.1 filter

`vtss_packet_filter_t vtss_packet_port_filter_t::filter`

Packet filtering

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

#### 5.84.2.2 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`

## 5.85 `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`

### 5.85.1 Detailed Description

Port info structure.

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

### 5.85.2 Field Documentation

#### 5.85.2.1 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`.

#### 5.85.2.2 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`.

#### 5.85.2.3 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`.

#### 5.85.2.4 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.

#### 5.85.2.5 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`

## 5.86 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]`

#### 5.86.1 Detailed Description

CPU Rx configuration.

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

#### 5.86.2 Field Documentation

### 5.86.2.1 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.

### 5.86.2.2 reg

```
vtss_packet_rx_reg_t vtss_packet_rx_conf_t::reg
```

Packet registration

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

### 5.86.2.3 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.

### 5.86.2.4 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

## 5.87 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`

### 5.87.1 Detailed Description

System frame header describing received frame.

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

### 5.87.2 Field Documentation

#### 5.87.2.1 `length`

`u32 vtss_packet_rx_header_t::length`

Frame length excluding CRC

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

#### 5.87.2.2 `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`.

#### 5.87.2.3 `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`.

#### 5.87.2.4 learn

`BOOL vtss_packet_rx_header_t::learn`

TRUE if learn frame

Definition at line 221 of file vtss\_packet\_api.h.

#### 5.87.2.5 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.

#### 5.87.2.6 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.

#### 5.87.2.7 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](#)

## 5.88 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`

### 5.88.1 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`.

### 5.88.2 Field Documentation

### 5.88.2.1 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`.

### 5.88.2.2 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`.

### 5.88.2.3 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`.

#### 5.88.2.4 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.

#### 5.88.2.5 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.

### 5.88.2.6 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.

### 5.88.2.7 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.

### 5.88.2.8 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.

### 5.88.2.9 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.

### 5.88.2.10 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.

### 5.88.2.11 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.

### 5.88.2.12 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.

### 5.88.2.13 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.

### 5.88.2.14 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`.

### 5.88.2.15 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`.

### 5.88.2.16 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`.

### 5.88.2.17 sflow\_type

```
vtss_sf_low_type_t vtss_packet_rx_info_t::sf_low_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: [sf\\_low\\_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, [sf\\_low\\_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.

### 5.88.2.18 sf\_low\_port\_no

```
vtss_port_no_t vtss_packet_rx_info_t::sf_low_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 [sf\\_low\\_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.

### 5.88.2.19 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.

### 5.88.2.20 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.

### 5.88.2.21 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.

### 5.88.2.22 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

## 5.89 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`

### 5.89.1 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.

## 5.89.2 Field Documentation

### 5.89.2.1 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.

### 5.89.2.2 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.

### 5.89.2.3 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.

### 5.89.2.4 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.

### 5.89.2.5 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 (sfflow-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`.

### 5.89.2.6 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`.

### 5.89.2.7 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](#)

## 5.90 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]

### 5.90.1 Detailed Description

Packet registration per port.

Definition at line 764 of file types.h.

## 5.90.2 Field Documentation

### 5.90.2.1 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.

### 5.90.2.2 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`

## 5.91 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`

### 5.91.1 Detailed Description

CPU Rx queue configuration.

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

### 5.91.2 Field Documentation

### 5.91.2.1 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.

### 5.91.2.2 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`

## 5.92 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`

### 5.92.1 Detailed Description

CPU Rx queue map.

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

## 5.92.2 Field Documentation

### 5.92.2.1 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.

### 5.92.2.2 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.

### 5.92.2.3 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.

### 5.92.2.4 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.

### 5.92.2.5 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.

### 5.92.2.6 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`.

### 5.92.2.7 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`.

### 5.92.2.8 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`.

### 5.92.2.9 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`.

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

- `vtss_api/include/vtss_packet_api.h`

## 5.93 vtss\_packet\_rx\_queue\_npi\_conf\_t Struct Reference

CPU Rx queue NPI configuration.

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

## Data Fields

- `BOOL enable`

### 5.93.1 Detailed Description

CPU Rx queue NPI configuration.

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

### 5.93.2 Field Documentation

#### 5.93.2.1 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`

## 5.94 `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`

### 5.94.1 Detailed Description

CPU Rx packet registration.

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

## 5.94.2 Field Documentation

### 5.94.2.1 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.

### 5.94.2.2 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.

### 5.94.2.3 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.

### 5.94.2.4 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.

#### 5.94.2.5 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`

## 5.95 `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]`

#### 5.95.1 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`.

#### 5.95.2 Field Documentation

##### 5.95.2.1 length

`u32 vtss_packet_tx_ifh_t::length`

Length of compiled IFH (in bytes)

Definition at line 2073 of file `vtss_packet_api.h`.

### 5.95.2.2 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

## 5.96 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](#)

### 5.96.1 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.

## 5.96.2 Field Documentation

### 5.96.2.1 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`.

### 5.96.2.2 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.

### 5.96.2.3 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.

### 5.96.2.4 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.

### 5.96.2.5 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.

### 5.96.2.6 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.

### 5.96.2.7 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.

#### 5.96.2.8 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.

#### 5.96.2.9 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.

#### 5.96.2.10 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.

#### 5.96.2.11 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.

#### 5.96.2.12 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.

### 5.96.2.13 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.

### 5.96.2.14 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.

#### 5.96.2.15 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.

#### 5.96.2.16 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.

#### 5.96.2.17 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.

#### 5.96.2.18 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.

#### 5.96.2.19 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.

#### 5.96.2.20 pdu\_offset

```
u32 vtss_packet_tx_info_t::pdu_offset
```

PDU offset in 8 bit word counts. Used in ptpt-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](#)

## 5.97 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](#)

#### 5.97.1 Detailed Description

10G Phy APC configuration

Definition at line 241 of file vtss\_phy\_10g\_api.h.

#### 5.97.2 Field Documentation

##### 5.97.2.1 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.

### 5.97.2.2 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`

## 5.98 vtss\_phy\_10g\_apc\_status\_t Struct Reference

10G Phy APC status

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

### Data Fields

- `BOOL reset`
- `BOOL freeze`

#### 5.98.1 Detailed Description

10G Phy APC status

Definition at line 247 of file vtss\_phy\_10g\_api.h.

#### 5.98.2 Field Documentation

##### 5.98.2.1 reset

`BOOL vtss_phy_10g_apc_status_t::reset`

APC reset status

Definition at line 248 of file vtss\_phy\_10g\_api.h.

### 5.98.2.2 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`

## 5.99 `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`

### 5.99.1 Detailed Description

10G PHY Automatic Failover configuration

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

### 5.99.2 Field Documentation

#### 5.99.2.1 `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`.

**5.99.2.2 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.

**5.99.2.3 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.

**5.99.2.4 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.

**5.99.2.5 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.

**5.99.2.6 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.

### 5.99.2.7 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`.

### 5.99.2.8 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`.

### 5.99.2.9 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`.

### 5.99.2.10 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`

## 5.100 `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`

### 5.100.1 Detailed Description

10G Phy Base KR Autoneg config

Definition at line 1207 of file `vtss_phy_10g_api.h`.

### 5.100.2 Field Documentation

#### 5.100.2.1 `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`.

#### 5.100.2.2 `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`.

#### 5.100.2.3 `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`

## 5.101 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

#### 5.101.1 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.

#### 5.101.2 Field Documentation

##### 5.101.2.1 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.

##### 5.101.2.2 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.

### 5.101.2.3 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.

### 5.101.2.4 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.

### 5.101.2.5 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.

### 5.101.2.6 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.

### 5.101.2.7 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](#)

## 5.102 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

#### 5.102.1 Detailed Description

10G Phy Base Link Advertisement capability config

Definition at line 1223 of file vtss\_phy\_10g\_api.h.

#### 5.102.2 Field Documentation

##### 5.102.2.1 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.

##### 5.102.2.2 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.

### 5.102.2.3 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.

### 5.102.2.4 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](#)

## 5.103 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

### 5.103.1 Detailed Description

10G Phy Base KR Training & Autoneg status

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

### 5.103.2 Field Documentation

### 5.103.2.1 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.

### 5.103.2.2 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.

### 5.103.2.3 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

## 5.104 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

### 5.104.1 Detailed Description

10G Phy Base KR Training & Autoneg config

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

### 5.104.2 Field Documentation

#### 5.104.2.1 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.

#### 5.104.2.2 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.

#### 5.104.2.3 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.

#### 5.104.2.4 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.

### 5.104.2.5 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`

## 5.105 `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`

### 5.105.1 Detailed Description

10G Phy Base KR Training config

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

### 5.105.2 Field Documentation

#### 5.105.2.1 enable

`BOOL vtss_phy_10g_base_kr_training_t::enable`

Enable KR training

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

### 5.105.2.2 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.

### 5.105.2.3 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.

### 5.105.2.4 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.

### 5.105.2.5 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](#)

## 5.106 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`

### 5.106.1 Detailed Description

10G Phy CKOUT config data

Malibu Only

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

### 5.106.2 Field Documentation

#### 5.106.2.1 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`.

#### 5.106.2.2 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`.

#### 5.106.2.3 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`.

#### 5.106.2.4 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.

#### 5.106.2.5 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.

#### 5.106.2.6 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](#)

## 5.107 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`

### 5.107.1 Detailed Description

Advertisement control data for Clause 37 aneg.

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

### 5.107.2 Field Documentation

#### 5.107.2.1 fdx

`BOOL vtss_phy_10g_clause_37_adv_t::fdx`

(FD)

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

#### 5.107.2.2 hdx

`BOOL vtss_phy_10g_clause_37_adv_t::hdx`

(HD) ,Not supported

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

#### 5.107.2.3 symmetric\_pause

`BOOL vtss_phy_10g_clause_37_adv_t::symmetric_pause`

(PS1)

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

#### 5.107.2.4 asymmetric\_pause

`BOOL vtss_phy_10g_clause_37_adv_t::asymmetric_pause`

(PS2)

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

### 5.107.2.5 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.

### 5.107.2.6 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.

### 5.107.2.7 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`

## 5.108 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`

### 5.108.1 Detailed Description

Clause 37 Auto-negotiation status for line and host.

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

## 5.108.2 Field Documentation

### 5.108.2.1 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`.

### 5.108.2.2 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`

## 5.109 `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`

### 5.109.1 Detailed Description

Clause 37 control struct.

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

## 5.109.2 Field Documentation

### 5.109.2.1 enable

`BOOL vtss_phy_10g_clause_37_control_t::enable`

Enable of Autoneg

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

### 5.109.2.2 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.

### 5.109.2.3 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.

### 5.109.2.4 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.

### 5.109.2.5 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.

### 5.109.2.6 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`

## 5.110 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`

### 5.110.1 Detailed Description

Clause 37 Auto-negotiation status.

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

### 5.110.2 Field Documentation

#### 5.110.2.1 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.

### 5.110.2.2 complete

`BOOL vtss_phy_10g_clause_37_status_t::complete`

Aneg completion status

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

### 5.110.2.3 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.

### 5.110.2.4 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`

## 5.111 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`

### 5.111.1 Detailed Description

10G Phy CLOCK Source Selection

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

### 5.111.2 Field Documentation

#### 5.111.2.1 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`

## 5.112 `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`

#### 5.112.1 Detailed Description

10G Phy Sublayer counters

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

### 5.112.2 Field Documentation

#### 5.112.2.1 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`

## 5.113 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`

#### 5.113.1 Detailed Description

10G OOS workaround options

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

#### 5.113.2 Field Documentation

##### 5.113.2.1 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.

##### 5.113.2.2 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.

### 5.113.2.3 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](#)

## 5.114 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](#)

### 5.114.1 Detailed Description

Firmware status.

Definition at line 2741 of file vtss\_phy\_10g\_api.h.

### 5.114.2 Field Documentation

#### 5.114.2.1 edc\_fw\_rev

```
u16 vtss_phy_10g_fw_status_t::edc_fw_rev
```

FW revision

Definition at line 2742 of file vtss\_phy\_10g\_api.h.

#### 5.114.2.2 edc\_fw\_chksum

`BOOL vtss_phy_10g_fw_status_t::edc_fw_chksum`

FW checksum. Fail=0, Pass=1

Definition at line 2743 of file vtss\_phy\_10g\_api.h.

#### 5.114.2.3 icpu\_activity

`BOOL vtss_phy_10g_fw_status_t::icpu_activity`

iCPU activity. Not Running=0, Running=1

Definition at line 2744 of file vtss\_phy\_10g\_api.h.

#### 5.114.2.4 edc\_fw\_api\_load

`BOOL vtss_phy_10g_fw_status_t::edc_fw_api_load`

EDC FW is loaded through API No=0, Yes=1

Definition at line 2745 of file vtss\_phy\_10g\_api.h.

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

- `vtss_api/include/vtss_phy_10g_api.h`

## 5.115 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`

### 5.115.1 Detailed Description

10G Phy Host clock config data

Malibu Only

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

### 5.115.2 Field Documentation

#### 5.115.2.1 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.

#### 5.115.2.2 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.

#### 5.115.2.3 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`

## 5.116 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`

### 5.116.1 Detailed Description

10G Phy IB configuration

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

### 5.116.2 Field Documentation

#### 5.116.2.1 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.

#### 5.116.2.2 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.

### 5.116.2.3 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.

### 5.116.2.4 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.

### 5.116.2.5 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.

### 5.116.2.6 agc

`ib_par_cfg vtss_phy_10g_ib_conf_t::agc`

AGC value

Definition at line 219 of file vtss\_phy\_10g\_api.h.

### 5.116.2.7 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.

### 5.116.2.8 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.

### 5.116.2.9 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.

### 5.116.2.10 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.

### 5.116.2.11 ld

`u8` vtss\_phy\_10g\_ib\_conf\_t::ld

level detect

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

### 5.116.2.12 prbs

`u8` vtss\_phy\_10g\_ib\_conf\_t::prbs

PRBS RX pattern selected

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

### 5.116.2.13 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`.

### 5.116.2.14 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`.

### 5.116.2.15 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`.

### 5.116.2.16 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`.

### 5.116.2.17 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`

## 5.117 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`

#### 5.117.1 Detailed Description

10G Phy IB configuration

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

#### 5.117.2 Field Documentation

##### 5.117.2.1 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.

##### 5.117.2.2 sig\_det

```
BOOL vtss_phy_10g_ib_status_t::sig_det
```

Signal detect

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

### 5.117.2.3 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](#)

## 5.118 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]`

### 5.118.1 Detailed Description

VSCOPE fast scan storage.

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

### 5.118.2 Field Documentation

#### 5.118.2.1 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.

### 5.118.2.2 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](#)

## 5.119 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](#)

### 5.119.1 Detailed Description

10G Phy part number and revision

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

### 5.119.2 Field Documentation

#### 5.119.2.1 part\_number

```
u16 vtss_phy_10g_id_t::part_number
```

Part number (Hex)

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

### 5.119.2.2 revision

`u16 vtss_phy_10g_id_t::revision`

Chip revision

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

### 5.119.2.3 channel\_id

`u16 vtss_phy_10g_id_t::channel_id`

Channel id

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

### 5.119.2.4 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.

### 5.119.2.5 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.

### 5.119.2.6 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.

### 5.119.2.7 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`

## 5.120 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`

### 5.120.1 Detailed Description

10G Phy Initialization configuration

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

### 5.120.2 Field Documentation

#### 5.120.2.1 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`

## 5.121 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`

#### 5.121.1 Detailed Description

10G Phy Optimisation of jitter performance

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

#### 5.121.2 Field Documentation

##### 5.121.2.1 incr\_levn

```
BOOL vtss_phy_10g_jitter_conf_t::incr_levn
```

Increase LevN

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

##### 5.121.2.2 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.

### 5.121.2.3 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](#)

## 5.122 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](#)

### 5.122.1 Detailed Description

10G Phy Base KR Autoneg status

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

### 5.122.2 Field Documentation

#### 5.122.2.1 complete

```
BOOL vtss_phy_10g_kr_status_aneg_t::complete
```

Aneg completed successfully

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

### 5.122.2.2 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.

### 5.122.2.3 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.

### 5.122.2.4 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.

### 5.122.2.5 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.

### 5.122.2.6 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.

### 5.122.2.7 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.

### 5.122.2.8 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.

### 5.122.2.9 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`

## 5.123 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`

### 5.123.1 Detailed Description

10G Phy Base KR FEC status

Definition at line 1261 of file vtss\_phy\_10g\_api.h.

### 5.123.2 Field Documentation

#### 5.123.2.1 enable

`BOOL vtss_phy_10g_kr_status_fec_t::enable`

FEC Enabled

Definition at line 1262 of file `vtss_phy_10g_api.h`.

#### 5.123.2.2 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`.

#### 5.123.2.3 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`

## 5.124 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`

### 5.124.1 Detailed Description

10G Phy Base KR Training status

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

### 5.124.2 Field Documentation

#### 5.124.2.1 complete

```
BOOL vtss_phy_10g_kr_status_train_t::complete
```

Training completed successfully

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

#### 5.124.2.2 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.

#### 5.124.2.3 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.

#### 5.124.2.4 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](#)

## 5.125 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`

#### 5.125.1 Detailed Description

10G Phy Lane SYNC Configuration

Malibu Only

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

#### 5.125.2 Field Documentation

##### 5.125.2.1 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.

##### 5.125.2.2 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.

### 5.125.2.3 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.

### 5.125.2.4 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.

### 5.125.2.5 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

## 5.126 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`

### 5.126.1 Detailed Description

10G Phy Line clock config data

Malibu Only

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

### 5.126.2 Field Documentation

#### 5.126.2.1 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.

#### 5.126.2.2 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.

#### 5.126.2.3 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`

## 5.127 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`

### 5.127.1 Detailed Description

10G Phy system and network loopbacks

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

### 5.127.2 Field Documentation

#### 5.127.2.1 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`.

#### 5.127.2.2 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`

## 5.128 **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`

### 5.128.1 Detailed Description

10G Phy operating mode

Definition at line 333 of file `vtss_phy_10g_api.h`.

### 5.128.2 Member Enumeration Documentation

#### 5.128.2.1 anonymous enum

anonymous enum

EDC modes.

Enumerator

<code>VTSS_EDC_FW_LOAD_MDIO</code>	Load EDC FW through MDIO to iCPU
<code>VTSS_EDC_FW_LOAD_NOTHING</code>	Do not load FW to iCPU

Definition at line 357 of file `vtss_phy_10g_api.h`.

### 5.128.3 Field Documentation

#### 5.128.3.1 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`.

#### 5.128.3.2 interface

`vtss_phy_interface_mode vtss_phy_10g_mode_t::interface`

Interface mode.

Definition at line 336 of file `vtss_phy_10g_api.h`.

### 5.128.3.3 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`.

### 5.128.3.4 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`.

### 5.128.3.5 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`.

### 5.128.3.6 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`.

### 5.128.3.7 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`.

### 5.128.3.8 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.

### 5.128.3.9 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.

### 5.128.3.10 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.

### 5.128.3.11 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.

### 5.128.3.12 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.

### 5.128.3.13 edc\_fw\_load

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

EDC modes.

EDC Firmware load

### 5.128.3.14 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.

### 5.128.3.15 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.

### 5.128.3.16 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.

### 5.128.3.17 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.

### 5.128.3.18 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.

### 5.128.3.19 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.

### 5.128.3.20 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.

### 5.128.3.21 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.

### 5.128.3.22 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.

### 5.128.3.23 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.

### 5.128.3.24 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.

### 5.128.3.25 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.

### 5.128.3.26 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.

### 5.128.3.27 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.

### 5.128.3.28 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\_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 380 of file vtss\_phy\_10g\_api.h.

### 5.128.3.29 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.

### 5.128.3.30 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.

### 5.128.3.31 serdes\_conf

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

Serdes parameters.

Serdes configuration

### 5.128.3.32 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.

### 5.128.3.33 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.

### 5.128.3.34 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.

### 5.128.3.35 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.

### 5.128.3.36 master

`clk_mstr_t vtss_phy_10g_mode_t::master`

Clock Master

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

### 5.128.3.37 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.

**5.128.3.38 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.

**5.128.3.39 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.

**5.128.3.40 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.

**5.128.3.41 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.

**5.128.3.42 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.

### 5.128.3.43 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.

### 5.128.3.44 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.

### 5.128.3.45 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.

### 5.128.3.46 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.

### 5.128.3.47 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.

**5.128.3.48 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`.

**5.128.3.49 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`.

**5.128.3.50 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`.

**5.128.3.51 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`.

**5.128.3.52 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`

## 5.129 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`

### 5.129.1 Detailed Description

10G Phy OB status

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

### 5.129.2 Field Documentation

#### 5.129.2.1 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.

#### 5.129.2.2 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.

### 5.129.2.3 slew

`u8 vtss_phy_10g_ob_status_t::slew`

slew rate

Definition at line 1174 of file vtss\_phy\_10g\_api.h.

### 5.129.2.4 levn

`u8 vtss_phy_10g_ob_status_t::levn`

amplitude

Definition at line 1175 of file vtss\_phy\_10g\_api.h.

### 5.129.2.5 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.

### 5.129.2.6 v3

`int vtss_phy_10g_ob_status_t::v3`

d\_filter tap v3

Definition at line 1177 of file vtss\_phy\_10g\_api.h.

### 5.129.2.7 vp

`int vtss_phy_10g_ob_status_t::vp`

d\_filter tap vp

Definition at line 1178 of file vtss\_phy\_10g\_api.h.

### 5.129.2.8 v4

```
int vtss_phy_10g_ob_status_t::v4
```

d\_filter tap v4

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

### 5.129.2.9 v5

```
int vtss_phy_10g_ob_status_t::v5
```

d\_filter tap v5

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

### 5.129.2.10 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](#)

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

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

### Data Fields

- `BOOL prbs_gen`

### 5.130.1 Detailed Description

\ brief 10G PHY pcs prbs generator configuration

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

## 5.130.2 Field Documentation

### 5.130.2.1 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`

## 5.131 vtss\_phy\_10g\_pcs\_prbs\_mon\_conf\_t Struct Reference

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

### Data Fields

- `BOOL prbs_mon`
- `u32 error_counter`

### 5.131.1 Detailed Description

\ brief 10G PHY pcs prbs analyzer configuration

Definition at line 1976 of file vtss\_phy\_10g\_api.h.

### 5.131.2 Field Documentation

#### 5.131.2.1 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.

### 5.131.2.2 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](#)

## 5.132 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](#)

### 5.132.1 Detailed Description

10G PHY Packet generator configuration

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

### 5.132.2 Field Documentation

### 5.132.2.1 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.

### 5.132.2.2 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.

### 5.132.2.3 ingress

`BOOL vtss_phy_10g_pkt_gen_conf_t::ingress`

Ingress or egress

Definition at line 2136 of file vtss\_phy\_10g\_api.h.

### 5.132.2.4 frames

`BOOL vtss_phy_10g_pkt_gen_conf_t::frames`

frames or idles

Definition at line 2137 of file vtss\_phy\_10g\_api.h.

### 5.132.2.5 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.

### 5.132.2.6 etype

`u16 vtss_phy_10g_pkt_gen_conf_t::etype`

Ethertype

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

### 5.132.2.7 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.

### 5.132.2.8 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.

### 5.132.2.9 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.

### 5.132.2.10 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.

### 5.132.2.11 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.

### 5.132.2.12 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.

### 5.132.2.13 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

## 5.133 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`

### 5.133.1 Detailed Description

10G PHY Packet Monitor configuration

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

### 5.133.2 Field Documentation

#### 5.133.2.1 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.

#### 5.133.2.2 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.

#### 5.133.2.3 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.

#### 5.133.2.4 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.

### 5.133.2.5 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`.

### 5.133.2.6 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`.

### 5.133.2.7 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`.

### 5.133.2.8 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`

## 5.134 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`

### 5.134.1 Detailed Description

10G Phy Polarity inversion

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

### 5.134.2 Field Documentation

#### 5.134.2.1 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`.

#### 5.134.2.2 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`.

#### 5.134.2.3 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`.

#### 5.134.2.4 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`

## 5.135 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`

#### 5.135.1 Detailed Description

\ brief 10G PHY prbs generator configuration

Definition at line 2026 of file vtss\_phy\_10g\_api.h.

#### 5.135.2 Field Documentation

##### 5.135.2.1 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.

### 5.135.2.2 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`.

### 5.135.2.3 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`.

### 5.135.2.4 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`.

### 5.135.2.5 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`

## 5.136 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`

### 5.136.1 Detailed Description

10G PHY prbs monitor Configuration

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

### 5.136.2 Field Documentation

#### 5.136.2.1 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.

#### 5.136.2.2 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.

### 5.136.2.3 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.

### 5.136.2.4 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.

### 5.136.2.5 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.

### 5.136.2.6 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.

### 5.136.2.7 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.

### 5.136.2.8 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.

### 5.136.2.9 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.

### 5.136.2.10 error\_status

`u32 vtss_phy_10g_prbs_mon_conf_t::error_status`

Error stautus of PRBS

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

### 5.136.2.11 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.

### 5.136.2.12 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.

### 5.136.2.13 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.

### 5.136.2.14 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.

### 5.136.2.15 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.

### 5.136.2.16 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.

### 5.136.2.17 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.

### 5.136.2.18 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`

## 5.137 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`

### 5.137.1 Detailed Description

10G Phy RXCKOUT config data

Definition at line 706 of file vtss\_phy\_10g\_api.h.

### 5.137.2 Field Documentation

#### 5.137.2.1 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.

### 5.137.2.2 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`.

### 5.137.2.3 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`

## 5.138 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`

### 5.138.1 Detailed Description

10G Phy SCKOUT config data

Malibu Only

Definition at line 982 of file `vtss_phy_10g_api.h`.

### 5.138.2 Field Documentation

### 5.138.2.1 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.

### 5.138.2.2 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.

### 5.138.2.3 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.

### 5.138.2.4 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.

### 5.138.2.5 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](#)

## 5.139 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

### 5.139.1 Detailed Description

10G Phy SERDES status

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

### 5.139.2 Field Documentation

### 5.139.2.1 rcomp

`u8 vtss_phy_10g_serdes_status_t::rcomp`

Measured Resistor value

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

### 5.139.2.2 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.

### 5.139.2.3 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.

### 5.139.2.4 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.

### 5.139.2.5 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.

### 5.139.2.6 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.

### 5.139.2.7 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.

### 5.139.2.8 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.

### 5.139.2.9 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.

### 5.139.2.10 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.

**5.139.2.11 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.

**5.139.2.12 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.

**5.139.2.13 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.

**5.139.2.14 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.

**5.139.2.15 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.

**5.139.2.16 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.

**5.139.2.17 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.

**5.139.2.18 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.

**5.139.2.19 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.

**5.139.2.20 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.

**5.139.2.21 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.

**5.139.2.22 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.

**5.139.2.23 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.

**5.139.2.24 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.

**5.139.2.25 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.

### 5.139.2.26 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`.

### 5.139.2.27 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`.

### 5.139.2.28 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`.

### 5.139.2.29 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`.

### 5.139.2.30 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`

## 5.140 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`

#### 5.140.1 Detailed Description

10G Phy srefclk config data

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

#### 5.140.2 Field Documentation

##### 5.140.2.1 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.

##### 5.140.2.2 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](#)

## 5.141 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`

### 5.141.1 Detailed Description

10G Phy link and fault status for all sublayers

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

### 5.141.2 Field Documentation

#### 5.141.2.1 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`.

#### 5.141.2.2 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`.

#### 5.141.2.3 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`.

#### 5.141.2.4 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.

#### 5.141.2.5 hpcs

`vtss_sublayer_status_t vtss_phy_10g_status_t::hpcs`

Status for HOST PCS sublayer,pcs xauि in case of venice

Definition at line 1433 of file vtss\_phy\_10g\_api.h.

#### 5.141.2.6 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.

#### 5.141.2.7 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.

#### 5.141.2.8 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.

### 5.141.2.9 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.

### 5.141.2.10 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.

### 5.141.2.11 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](#)

## 5.142 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]`

### 5.142.1 Detailed Description

10G PHY timestamp value array(holder)

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

## 5.142.2 Field Documentation

### 5.142.2.1 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](#)

## 5.143 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](#)

### 5.143.1 Detailed Description

10G Phy TXCKOUT config data

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

## 5.143.2 Field Documentation

### 5.143.2.1 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](#)

## 5.144 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`

#### 5.144.1 Detailed Description

\ brief VSCOPE scan configuration

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

#### 5.144.2 Field Documentation

##### 5.144.2.1 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.

##### 5.144.2.2 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.

##### 5.144.2.3 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.

#### 5.144.2.4 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`

## 5.145 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`

#### 5.145.1 Detailed Description

VSCOPE scan configuration.

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

#### 5.145.2 Field Documentation

##### 5.145.2.1 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.

### 5.145.2.2 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.

### 5.145.2.3 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.

### 5.145.2.4 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.

### 5.145.2.5 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.

### 5.145.2.6 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.

### 5.145.2.7 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.

### 5.145.2.8 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

## 5.146 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 \[PHASE\\_POINTS\]\[AMPLITUDE\\_POINTS\]](#)

### 5.146.1 Detailed Description

\ brief Vscope eye scan status

Definition at line 1919 of file vtss\_phy\_10g\_api.h.

### 5.146.2 Field Documentation

### 5.146.2.1 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`.

### 5.146.2.2 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`.

### 5.146.2.3 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`.

### 5.146.2.4 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`.

### 5.146.2.5 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`

## 5.147 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`

### 5.147.1 Detailed Description

PHY auto negotiation advertisement.

Definition at line 309 of file vtss\_phy\_api.h.

### 5.147.2 Field Documentation

#### 5.147.2.1 speed\_10m\_hdx

```
BOOL vtss_phy_aneg_t::speed_10m_hdx
```

10Mbps, half duplex

Definition at line 310 of file vtss\_phy\_api.h.

#### 5.147.2.2 speed\_10m\_fdx

```
BOOL vtss_phy_aneg_t::speed_10m_fdx
```

10Mbps, full duplex

Definition at line 311 of file vtss\_phy\_api.h.

### 5.147.2.3 speed\_100m\_hdx

`BOOL vtss_phy_aneg_t::speed_100m_hdx`

100Mbps, half duplex

Definition at line 312 of file `vtss_phy_api.h`.

### 5.147.2.4 speed\_100m\_fdx

`BOOL vtss_phy_aneg_t::speed_100m_fdx`

100Mbps, full duplex

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

### 5.147.2.5 speed\_1g\_fdx

`BOOL vtss_phy_aneg_t::speed_1g_fdx`

1000Mbps, full duplex

Definition at line 314 of file `vtss_phy_api.h`.

### 5.147.2.6 speed\_1g\_hdx

`BOOL vtss_phy_aneg_t::speed_1g_hdx`

1000Mbps, full duplex

Definition at line 315 of file `vtss_phy_api.h`.

### 5.147.2.7 symmetric\_pause

`BOOL vtss_phy_aneg_t::symmetric_pause`

Symmetric pause

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

### 5.147.2.8 asymmetric\_pause

`BOOL vtss_phy_aneg_t::asymmetric_pause`

Asymmetric pause

Definition at line 317 of file vtss\_phy\_api.h.

### 5.147.2.9 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](#)

## 5.148 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](#)

### 5.148.1 Detailed Description

PHY clock configuration.

Definition at line 648 of file vtss\_phy\_api.h.

### 5.148.2 Field Documentation

### 5.148.2.1 src

```
vtss_phy_clk_source_t vtss_phy_clock_conf_t::src
```

Clock source

Definition at line 649 of file vtss\_phy\_api.h.

### 5.148.2.2 freq

```
vtss_phy_freq_t vtss_phy_clock_conf_t::freq
```

Clock frequency

Definition at line 650 of file vtss\_phy\_api.h.

### 5.148.2.3 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

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

PHY 1G configuration.

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

### Data Fields

- struct {  
    BOOL cfg  
    BOOL val  
} master

### 5.149.1 Detailed Description

PHY 1G configuration.

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

### 5.149.2 Field Documentation

#### 5.149.2.1 cfg

`BOOL vtss_phy_conf_1g_t::cfg`

Manual Master/Slave Config. 1=enabled

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

#### 5.149.2.2 val

`BOOL vtss_phy_conf_1g_t::val`

Master/Slave Config value, 1=Master

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

#### 5.149.2.3 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](#)

## 5.150 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`

### 5.150.1 Detailed Description

PHY configuration.

Definition at line 396 of file `vtss_phy_api.h`.

### 5.150.2 Field Documentation

#### 5.150.2.1 mode

`vtss_phy_mode_t vtss_phy_conf_t::mode`

PHY mode

Definition at line 397 of file `vtss_phy_api.h`.

#### 5.150.2.2 forced

`vtss_phy_forced_t vtss_phy_conf_t::forced`

Forced mode configuration

Definition at line 398 of file `vtss_phy_api.h`.

#### 5.150.2.3 aneg

`vtss_phy_aneg_t vtss_phy_conf_t::aneg`

Auto-negotiation mode configuration

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

#### 5.150.2.4 mdi

`vtss_phy_mdi_t` `vtss_phy_conf_t::mdi`

Cu cable MDI (Crossed cable / normal cable)

Definition at line 400 of file vtss\_phy\_api.h.

#### 5.150.2.5 flf

`vtss_phy_fast_link_fail_t` `vtss_phy_conf_t::flf`

Fast link failure configuration

Definition at line 401 of file vtss\_phy\_api.h.

#### 5.150.2.6 sigdet

`vtss_phy_sigdet_polarity_t` `vtss_phy_conf_t::sigdet`

Sigdet pin polarity configuration

Definition at line 402 of file vtss\_phy\_api.h.

#### 5.150.2.7 unidir

`vtss_phy_unidirectional_t` `vtss_phy_conf_t::unidir`

Unidirectional Configuration

Definition at line 403 of file vtss\_phy\_api.h.

#### 5.150.2.8 mac\_if\_pcs

`vtss_phy_mac_serdes_pcs_ctrl_t` `vtss_phy_conf_t::mac_if_pcs`

PHY MAC SerDes PCS Control (Reg16E3)

Definition at line 404 of file vtss\_phy\_api.h.

### 5.150.2.9 media\_if\_pcs

`vtss_phy_media_serdes_pcs_cntl_t` `vtss_phy_conf_t::media_if_pcs`

PHY MAC SerDes PCS Control (Reg23E3)

Definition at line 405 of file `vtss_phy_api.h`.

### 5.150.2.10 force\_ams\_sel

`vtss_phy_media_force_ams_sel_t` `vtss_phy_conf_t::force_ams_sel`

PHY Media AMS Force Selection

Definition at line 406 of file `vtss_phy_api.h`.

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

- `vtss_api/include/vtss_phy_api.h`

## 5.151 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`

### 5.151.1 Detailed Description

SPI daisy chain configuration.

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

### 5.151.2 Field Documentation

### 5.151.2.1 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`.

### 5.151.2.2 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`

## 5.152 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`

### 5.152.1 Detailed Description

EEE configuration.

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

### 5.152.2 Field Documentation

### 5.152.2.1 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`.

### 5.152.2.2 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`

## 5.153 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`

### 5.153.1 Detailed Description

enhanced LED control

Definition at line 1010 of file `vtss_phy_api.h`.

### 5.153.2 Field Documentation

### 5.153.2.1 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.

### 5.153.2.2 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.

### 5.153.2.3 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.

### 5.153.2.4 ser\_led\_select

```
u8 vtss_phy_enhanced_led_control_t::ser_led_select
```

The number of LEDs the hardware board supports for each PHY, 0x00 = 4 LEDs, 0x01 = 3 LEDs, 0x02 = 2 LEDs, 0x03 = 1 LED

Definition at line 1014 of file vtss\_phy\_api.h.

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

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

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

PHY forced mode configuration.

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

## Data Fields

- `vtss_port_speed_t speed`
- `BOOL fdx`

### 5.154.1 Detailed Description

PHY forced mode configuration.

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

### 5.154.2 Field Documentation

#### 5.154.2.1 speed

`vtss_port_speed_t vtss_phy_forced_t::speed`

Speed

Definition at line 304 of file `vtss_phy_api.h`.

#### 5.154.2.2 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`

## 5.155 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`

### 5.155.1 Detailed Description

LED model selection.

Definition at line 136 of file vtss\_phy\_api.h.

### 5.155.2 Field Documentation

#### 5.155.2.1 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.

#### 5.155.2.2 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`

## 5.156 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`

### 5.156.1 Detailed Description

1G Phy loopbacks

Definition at line 1384 of file vtss\_phy\_api.h.

### 5.156.2 Field Documentation

#### 5.156.2.1 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.

#### 5.156.2.2 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.

#### 5.156.2.3 connector\_enable

```
BOOL vtss_phy_loopback_t::connector_enable
```

Enable/Disable loopback connector loopback

Definition at line 1387 of file vtss\_phy\_api.h.

#### 5.156.2.4 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.

### 5.156.2.5 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.

### 5.156.2.6 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.

### 5.156.2.7 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.

### 5.156.2.8 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.

### 5.156.2.9 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](#)

## 5.157 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`

#### 5.157.1 Detailed Description

Frequency synthesis pulse configuration.

Definition at line 501 of file vtss\_phy\_ts\_api.h.

#### 5.157.2 Field Documentation

##### 5.157.2.1 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.

##### 5.157.2.2 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.

### 5.157.2.3 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

## 5.158 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

### 5.158.1 Detailed Description

PHY MAC SerDes PCS Control, Reg16E3.

Definition at line 351 of file vtss\_phy\_api.h.

### 5.158.2 Field Documentation

### 5.158.2.1 disable

`BOOL vtss_phy_mac_serdesrd_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`.

### 5.158.2.2 restart

`BOOL vtss_phy_mac_serdesrd_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`.

### 5.158.2.3 pd\_enable

`BOOL vtss_phy_mac_serdesrd_pcs_ctrl_t::pd_enable`

MAC i/f ANEG parallel detect enable

Definition at line 354 of file `vtss_phy_api.h`.

### 5.158.2.4 aneg\_restart

`BOOL vtss_phy_mac_serdesrd_pcs_ctrl_t::aneg_restart`

Restart MAC i/f ANEG

Definition at line 355 of file `vtss_phy_api.h`.

### 5.158.2.5 force\_adv\_ability

`BOOL vtss_phy_mac_serdesrd_pcs_ctrl_t::force_adv_ability`

Force adv. ability from Reg18E3

Definition at line 356 of file `vtss_phy_api.h`.

### 5.158.2.6 sgmii\_in\_pre

```
vtss_phy_mac_serdes_pcs_sgmii_pre vtss_phy_mac_serdes_pcs_ctrl_t::sgmii_in_pre
```

SGMII Input Preamble for 100BaseFX

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

### 5.158.2.7 sgmii\_out\_pre

```
BOOL vtss_phy_mac_serdes_pcs_ctrl_t::sgmii_out_pre
```

SGMII Output Preamble

Definition at line 358 of file vtss\_phy\_api.h.

### 5.158.2.8 serdes\_aneg\_ena

```
BOOL vtss_phy_mac_serdes_pcs_ctrl_t::serdes_aneg_ena
```

MAC SerDes ANEG Enable

Definition at line 359 of file vtss\_phy\_api.h.

### 5.158.2.9 serdes\_pol\_inv\_in

```
BOOL vtss_phy_mac_serdes_pcs_ctrl_t::serdes_pol_inv_in
```

Invert SerDes Polarity at input of MAC

Definition at line 360 of file vtss\_phy\_api.h.

### 5.158.2.10 serdes\_pol\_inv\_out

```
BOOL vtss_phy_mac_serdes_pcs_ctrl_t::serdes_pol_inv_out
```

Invert SerDes Polarity at output of MAC

Definition at line 361 of file vtss\_phy\_api.h.

### 5.158.2.11 fast\_link\_stat\_ena

`BOOL vtss_phy_mac_serdes_pcs_cntl_t::fast_link_stat_ena`

Fast Link Fail Status Enable

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

### 5.158.2.12 inhibit\_odd\_start

`BOOL vtss_phy_mac_serdes_pcs_cntl_t::inhibit_odd_start`

Inhibit MAC Odd-Start delay

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

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

- `vtss_api/include/vtss_phy_api.h`

## 5.159 vtss\_phy\_media\_serdes\_pcs\_cntl\_t Struct Reference

PHY MEDIA SerDes PCS Control, Reg23E3.

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

### Data Fields

- `vtss_phy_media_rem_fault_t remote_fault`
- `BOOL aneg_pd_detect`
- `BOOL force_adv_ability`
- `BOOL serdes_pol_inv_in`
- `BOOL serdes_pol_inv_out`
- `BOOL inhibit_odd_start`
- `BOOL force_hls`
- `BOOL force_fefi`
- `BOOL force_fefi_value`

### 5.159.1 Detailed Description

PHY MEDIA SerDes PCS Control, Reg23E3.

Definition at line 375 of file `vtss_phy_api.h`.

## 5.159.2 Field Documentation

### 5.159.2.1 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.

### 5.159.2.2 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.

### 5.159.2.3 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.

### 5.159.2.4 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.

### 5.159.2.5 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.

### 5.159.2.6 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.

### 5.159.2.7 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.

### 5.159.2.8 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.

### 5.159.2.9 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](#)

## 5.160 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`

### 5.160.1 Detailed Description

10G Phy PCS counters

Definition at line 1668 of file `vtss_phy_10g_api.h`.

### 5.160.2 Field Documentation

#### 5.160.2.1 `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`.

#### 5.160.2.2 `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`.

#### 5.160.2.3 `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`.

#### 5.160.2.4 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](#)

### 5.161 vtss\_phy\_power\_conf\_t Struct Reference

PHY power configuration.

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

#### Data Fields

- [vtss\\_phy\\_power\\_mode\\_t mode](#)

#### 5.161.1 Detailed Description

PHY power configuration.

Definition at line 562 of file vtss\_phy\_api.h.

#### 5.161.2 Field Documentation

##### 5.161.2.1 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](#)

## 5.162 vtss\_phy\_power\_status\_t Struct Reference

PHY power status.

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

### Data Fields

- [u32 level](#)

#### 5.162.1 Detailed Description

PHY power status.

Definition at line 597 of file vtss\_phy\_api.h.

#### 5.162.2 Field Documentation

##### 5.162.2.1 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](#)

## 5.163 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](#)

### 5.163.1 Detailed Description

PHY reset structure.

Definition at line 218 of file vtss\_phy\_api.h.

### 5.163.2 Field Documentation

#### 5.163.2.1 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.

#### 5.163.2.2 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.

#### 5.163.2.3 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.

#### 5.163.2.4 tbi

`vtss_phy_tbi_conf_t` `vtss_phy_reset_conf_t::tbi`

TBI setup

Definition at line 222 of file vtss\_phy\_api.h.

### 5.163.2.5 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`.

### 5.163.2.6 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`.

### 5.163.2.7 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`

## 5.164 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`

### 5.164.1 Detailed Description

PHY RGMII configuration.

Definition at line 194 of file `vtss_phy_api.h`.

## 5.164.2 Field Documentation

### 5.164.2.1 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.

### 5.164.2.2 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](#)

## 5.165 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

### 5.165.1 Detailed Description

Phy statistic information.

Definition at line 1348 of file vtss\_phy\_api.h.

## 5.165.2 Field Documentation

### 5.165.2.1 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.

### 5.165.2.2 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.

### 5.165.2.3 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.

### 5.165.2.4 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.

### 5.165.2.5 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`.

### 5.165.2.6 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`.

### 5.165.2.7 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`

## 5.166 vtss\_phy\_status\_1g\_t Struct Reference

PHY 1G status.

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

### Data Fields

- `BOOL master_cfg_fault`
- `BOOL master`

### 5.166.1 Detailed Description

PHY 1G status.

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

### 5.166.2 Field Documentation

#### 5.166.2.1 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.

#### 5.166.2.2 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`

## 5.167 vtss\_phy\_tbi\_conf\_t Struct Reference

PHY TBI configuration.

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

### Data Fields

- `BOOL aneg_enable`

### 5.167.1 Detailed Description

PHY TBI configuration.

Definition at line 200 of file vtss\_phy\_api.h.

## 5.167.2 Field Documentation

### 5.167.2.1 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`

## 5.168 `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`

### 5.168.1 Detailed Description

PHY timestamp in seconds and nanoseconds (10 bytes Timestamp)

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

### 5.168.2 Field Documentation

#### 5.168.2.1 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`.

### 5.168.2.2 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.

### 5.168.2.3 seconds

`struct { ... } vtss_phy_timestamp_t::seconds`

6 bytes second part of Timestamp

### 5.168.2.4 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

## 5.169 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`

### 5.169.1 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.

### 5.169.2 Field Documentation

#### 5.169.2.1 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.

#### 5.169.2.2 mask [1/2]

```
u8 vtss_phy_ts_ach_conf_t::mask
```

Mask

Definition at line 1103 of file vtss\_phy\_ts\_api.h.

#### 5.169.2.3 version

```
struct { ... } vtss_phy_ts_ach_conf_t::version
```

version number of the PWACH in value/mask; set mask 0 for don't care

#### 5.169.2.4 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.

### 5.169.2.5 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.

### 5.169.2.6 channel\_type

`struct { ... } vtss_phy_ts_ach_conf_t::channel_type`

PW Associated Channel Type in value/mask format

### 5.169.2.7 proto\_id

`struct { ... } vtss_phy_ts_ach_conf_t::proto_id`

PID: identifier of payload as defined in RFC 5718, only for PTP

### 5.169.2.8 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`

## 5.170 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`

### 5.170.1 Detailed Description

parameter for setting the alternative clock mode.

external clock output configuration.

Definition at line 52 of file vtss\_phy\_ts\_api.h.

### 5.170.2 Field Documentation

#### 5.170.2.1 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.

#### 5.170.2.2 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.

#### 5.170.2.3 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](#)

## 5.171 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`

### 5.171.1 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`.

### 5.171.2 Field Documentation

#### 5.171.2.1 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`.

#### 5.171.2.2 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`.

#### 5.171.2.3 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`.

#### 5.171.2.4 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`.

#### 5.171.2.5 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`

## 5.172 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`

#### 5.172.1 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`.

## 5.172.2 Field Documentation

### 5.172.2.1 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.

### 5.172.2.2 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.

### 5.172.2.3 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.

### 5.172.2.4 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.

### 5.172.2.5 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.

### 5.172.2.6 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](#)

## 5.173 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`

### 5.173.1 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`.

### 5.173.2 Field Documentation

#### 5.173.2.1 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`.

### 5.173.2.2 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.

### 5.173.2.3 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.

### 5.173.2.4 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.

### 5.173.2.5 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.

### 5.173.2.6 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](#)

## 5.174 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]

### 5.174.1 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.

## 5.174.2 Field Documentation

### 5.174.2.1 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.

### 5.174.2.2 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.

### 5.174.2.3 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.

### 5.174.2.4 comm\_opt

`struct { ... } vtss_phy_ts_eth_conf_t::comm_opt`

Ethernet common config

### 5.174.2.5 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.

### 5.174.2.6 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.

### 5.174.2.7 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.

### 5.174.2.8 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.

### 5.174.2.9 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.

#### 5.174.2.10 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.

#### 5.174.2.11 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.

#### 5.174.2.12 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.

#### 5.174.2.13 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.

#### 5.174.2.14 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.

**5.174.2.15 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`.

**5.174.2.16 range [1/2]**

`struct { ... } vtss_phy_ts_eth_conf_t::range`

tag in range

**5.174.2.17 val [1/2]**

`u16 vtss_phy_ts_eth_conf_t::val`

Value

Definition at line 993 of file `vtss_phy_ts_api.h`.

**5.174.2.18 mask [1/2]**

`u16 vtss_phy_ts_eth_conf_t::mask`

Mask

Definition at line 994 of file `vtss_phy_ts_api.h`.

**5.174.2.19 value [1/2]**

`struct { ... } vtss_phy_ts_eth_conf_t::value`

tag in value/mask

**5.174.2.20 outer\_tag**

`union { ... } vtss_phy_ts_eth_conf_t::outer_tag`

Outer tag

**5.174.2.21 range [2/2]**

```
struct { ... } vtss_phy_ts_eth_conf_t::range
```

tag in range

**5.174.2.22 value [2/2]**

```
struct { ... } vtss_phy_ts_eth_conf_t::value
```

tag in value/mask

**5.174.2.23 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.

**5.174.2.24 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.

**5.174.2.25 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

**5.174.2.26 inner\_tag**

```
union { ... } vtss_phy_ts_eth_conf_t::inner_tag
```

Inner Tag

### 5.174.2.27 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](#)

## 5.175 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`

### 5.175.1 Detailed Description

Defines the params for FIFO SYNC function.

Definition at line 2141 of file `vtss_phy_ts_api.h`.

### 5.175.2 Field Documentation

#### 5.175.2.1 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`.

### 5.175.2.2 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.

### 5.175.2.3 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.

### 5.175.2.4 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

## 5.176 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 \[6\]](#)

### 5.176.1 Detailed Description

Tx TSFIFO entry signature.

Definition at line 669 of file vtss\_phy\_ts\_api.h.

### 5.176.2 Field Documentation

#### 5.176.2.1 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.

#### 5.176.2.2 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.

#### 5.176.2.3 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.

#### 5.176.2.4 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.

### 5.176.2.5 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.

### 5.176.2.6 dest\_ip

`u32 vtss_phy_ts_fifo_sig_t::dest_ip`

Destination IP

Definition at line 675 of file vtss\_phy\_ts\_api.h.

### 5.176.2.7 src\_ip

`u32 vtss_phy_ts_fifo_sig_t::src_ip`

Source IP

Definition at line 676 of file vtss\_phy\_ts\_api.h.

### 5.176.2.8 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](#)

## 5.177 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]

### 5.177.1 Detailed Description

Analyzer Generic data configuration options using IP comparator.

Definition at line 1122 of file vtss\_phy\_ts\_api.h.

### 5.177.2 Field Documentation

#### 5.177.2.1 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.

#### 5.177.2.2 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.

#### 5.177.2.3 comm\_opt

struct { ... } vtss\_phy\_ts\_gen\_conf\_t::comm\_opt

Generic Matching common configuration

#### 5.177.2.4 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`.

#### 5.177.2.5 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`.

#### 5.177.2.6 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`.

#### 5.177.2.7 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`

## 5.178 **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`

### 5.178.1 Detailed Description

Generic Action configuration option.

Definition at line 1400 of file `vtss_phy_ts_api.h`.

### 5.178.2 Field Documentation

#### 5.178.2.1 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`.

#### 5.178.2.2 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`.

#### 5.178.2.3 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`.

#### 5.178.2.4 data

`u32 vtss_phy_ts_generic_action_t::data[2]`

Matching data pattern

Definition at line 1404 of file `vtss_phy_ts_api.h`.

#### 5.178.2.5 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`.

#### 5.178.2.6 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`.

#### 5.178.2.7 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`

## 5.179 **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](#)

### 5.179.1 Detailed Description

Generic engine flow configuration options.

Definition at line 1168 of file vtss\_phy\_ts\_api.h.

### 5.179.2 Field Documentation

#### 5.179.2.1 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.

#### 5.179.2.2 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](#)

## 5.180 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](#)

### 5.180.1 Detailed Description

Analyzer OAM comparator, IETF MPLS ACH OAM Packet format configuration options.

Definition at line 1368 of file `vtss_phy_ts_api.h`.

### 5.180.2 Field Documentation

#### 5.180.2.1 `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`.

#### 5.180.2.2 `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`.

#### 5.180.2.3 `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`

## 5.181 `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`

### 5.181.1 Detailed Description

Defines the initial parameters to be passed to init function.

Definition at line 1743 of file vtss\_phy\_ts\_api.h.

### 5.181.2 Field Documentation

#### 5.181.2.1 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.

#### 5.181.2.2 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.

### 5.181.2.3 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.

### 5.181.2.4 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.

### 5.181.2.5 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.

### 5.181.2.6 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.

### 5.181.2.7 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.

**5.181.2.8 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.

**5.181.2.9 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.

**5.181.2.10 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.

**5.181.2.11 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.

**5.181.2.12 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.

**5.181.2.13 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`.

**5.181.2.14 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`.

**5.181.2.15 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`

---

**5.182 **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]

```

### 5.182.1 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.

### 5.182.2 Field Documentation

#### 5.182.2.1 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.

### 5.182.2.2 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.

### 5.182.2.3 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.

### 5.182.2.4 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.

### 5.182.2.5 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.

### 5.182.2.6 comm\_opt

```
struct { ... } vtss_phy_ts_ip_conf_t::comm_opt
```

IP common config

### 5.182.2.7 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.

### 5.182.2.8 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.

### 5.182.2.9 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.

### 5.182.2.10 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.

### 5.182.2.11 ipv4

`struct { ... } vtss_phy_ts_ip_conf_t::ipv4`

IPv4 Address

## 5.182.2.12 ipv6

```
struct { ... } vtss_phy_ts_ip_conf_t::ipv6
```

IPv6 Mask

## 5.182.2.13 ip\_addr

```
union { ... } vtss_phy_ts_ip_conf_t::ip_addr
```

IPv4/IPv6 address to be matched

## 5.182.2.14 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](#)

## 5.183 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]

### 5.183.1 Detailed Description

Analyzer MPLS comparator configuration options.

Definition at line 1061 of file vtss\_phy\_ts\_api.h.

### 5.183.2 Field Documentation

#### 5.183.2.1 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.

#### 5.183.2.2 comm\_opt

`struct { ... } vtss_phy_ts_mpls_conf_t::comm_opt`

MPLS common config

#### 5.183.2.3 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.

#### 5.183.2.4 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.

### 5.183.2.5 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.

### 5.183.2.6 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.

### 5.183.2.7 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.

### 5.183.2.8 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.

### 5.183.2.9 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.

### 5.183.2.10 top\_down

```
struct { ... } vtss_phy_ts_mpls_conf_t::top_down
```

Top down configuration

### 5.183.2.11 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.

### 5.183.2.12 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.

### 5.183.2.13 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.

### 5.183.2.14 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.

### 5.183.2.15 bottom\_up

```
struct { ... } vtss_phy_ts_mpls_conf_t::bottom_up
```

Bottom up configuration

### 5.183.2.16 stack\_level

```
union { ... } vtss_phy_ts_mpls_conf_t::stack_level
```

4 level values; top\_down or bottom\_up depends on stack\_ref\_point

### 5.183.2.17 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](#)

## 5.184 vtss\_phy\_ts\_mpls\_lvl\_rng\_t Struct Reference

MPLS level range.

```
#include <vtss_phy_ts_api.h>
```

### Data Fields

- [u32 lower](#)
- [u32 upper](#)

#### 5.184.1 Detailed Description

MPLS level range.

Definition at line 1053 of file [vtss\\_phy\\_ts\\_api.h](#).

#### 5.184.2 Field Documentation

##### 5.184.2.1 lower

```
u32 vtss_phy_ts_mpls_lvl_rng_t::lower
```

lower range value

Definition at line 1054 of file [vtss\\_phy\\_ts\\_api.h](#).

### 5.184.2.2 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`

## 5.185 `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`

### 5.185.1 Detailed Description

n-phase status

Definition at line 1885 of file `vtss_phy_ts_api.h`.

### 5.185.2 Field Documentation

#### 5.185.2.1 enable

`BOOL vtss_phy_ts_nphase_status_t::enable`

Enabled status

Definition at line 1886 of file `vtss_phy_ts_api.h`.

### 5.185.2.2 CALIB\_ERR

`BOOL vtss_phy_ts_nphase_status_t::CALIB_ERR`

Calibration error

Definition at line 1887 of file vtss\_phy\_ts\_api.h.

### 5.185.2.3 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`

## 5.186 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`

### 5.186.1 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.

## 5.186.2 Field Documentation

### 5.186.2.1 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.

### 5.186.2.2 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.

### 5.186.2.3 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.

### 5.186.2.4 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.

### 5.186.2.5 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.

### 5.186.2.6 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`.

### 5.186.2.7 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`

## 5.187 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`

### 5.187.1 Detailed Description

OAM engine flow configuration options.

Definition at line 1158 of file `vtss_phy_ts_api.h`.

### 5.187.2 Field Documentation

### 5.187.2.1 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`.

### 5.187.2.2 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`.

### 5.187.2.3 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`.

### 5.187.2.4 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`

## 5.188 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`

### 5.188.1 Detailed Description

PPS Configuration.

Definition at line 102 of file `vtss_phy_ts_api.h`.

### 5.188.2 Field Documentation

#### 5.188.2.1 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`.

#### 5.188.2.2 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`.

#### 5.188.2.3 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`

## 5.189 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`

### 5.189.1 Detailed Description

Analyzer PTP comparator configuration options.

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

### 5.189.2 Field Documentation

#### 5.189.2.1 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.

#### 5.189.2.2 val

```
u8 vtss_phy_ts_ptp_conf_t::val
```

PTP domain number value

Definition at line 1272 of file vtss\_phy\_ts\_api.h.

### 5.189.2.3 mask

```
u8 vtss_phy_ts_ptp_conf_t::mask
```

PTP domain number mask

Definition at line 1273 of file vtss\_phy\_ts\_api.h.

### 5.189.2.4 value

```
struct { ... } vtss_phy_ts_ptp_conf_t::value
```

specific PTP domain, for don't care set mask as '0'

### 5.189.2.5 upper

```
u8 vtss_phy_ts_ptp_conf_t::upper
```

Ranger upper value

Definition at line 1276 of file vtss\_phy\_ts\_api.h.

### 5.189.2.6 lower

```
u8 vtss_phy_ts_ptp_conf_t::lower
```

Range lower value

Definition at line 1277 of file vtss\_phy\_ts\_api.h.

### 5.189.2.7 range

```
struct { ... } vtss_phy_ts_ptp_conf_t::range
```

PTP domain range configuration

### 5.189.2.8 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

## 5.190 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`

### 5.190.1 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.

### 5.190.2 Field Documentation

#### 5.190.2.1 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.

#### 5.190.2.2 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.

### 5.190.2.3 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`.

### 5.190.2.4 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`.

### 5.190.2.5 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`.

### 5.190.2.6 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`

## 5.191 **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`](#)

### 5.191.1 Detailed Description

PTP engine flow configuration options.

Definition at line 1146 of file `vtss_phy_ts_api.h`.

### 5.191.2 Field Documentation

#### 5.191.2.1 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`.

#### 5.191.2.2 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`.

#### 5.191.2.3 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`.

### 5.191.2.4 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`.

### 5.191.2.5 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`.

### 5.191.2.6 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`

## 5.192 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`

### 5.192.1 Detailed Description

PHY timestamp in seconds and nanoseconds (10 bytes Timestamp)

Definition at line 330 of file `vtss_phy_ts_api.h`.

## 5.192.2 Field Documentation

### 5.192.2.1 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`.

### 5.192.2.2 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`.

### 5.192.2.3 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`

## 5.193 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`

### 5.193.1 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.

### 5.193.2 Field Documentation

#### 5.193.2.1 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.

#### 5.193.2.2 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.

#### 5.193.2.3 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.

#### 5.193.2.4 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`.

#### 5.193.2.5 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`.

#### 5.193.2.6 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`.

#### 5.193.2.7 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`.

#### 5.193.2.8 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`

## 5.194 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`

### 5.194.1 Detailed Description

Analyzer OAM comparator, Y.1731 OAM Packet format configuration options.

Definition at line 1342 of file vtss\_phy\_ts\_api.h.

### 5.194.2 Field Documentation

#### 5.194.2.1 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.

#### 5.194.2.2 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.

### 5.194.2.3 val

`u8 vtss_phy_ts_y1731_oam_conf_t::val`

Value

Definition at line 1347 of file vtss\_phy\_ts\_api.h.

### 5.194.2.4 mask

`u8 vtss_phy_ts_y1731_oam_conf_t::mask`

Mask

Definition at line 1348 of file vtss\_phy\_ts\_api.h.

### 5.194.2.5 value

`struct { ... } vtss_phy_ts_y1731_oam_conf_t::value`

specific MEG Level, for don't care set mask as '0'

### 5.194.2.6 upper

`u8 vtss_phy_ts_y1731_oam_conf_t::upper`

Range Upper value

Definition at line 1351 of file vtss\_phy\_ts\_api.h.

### 5.194.2.7 lower

`u8 vtss_phy_ts_y1731_oam_conf_t::lower`

Range lower value

Definition at line 1352 of file vtss\_phy\_ts\_api.h.

### 5.194.2.8 range

`struct { ... } vtss_phy_ts_y1731_oam_conf_t::range`

Range configuration

### 5.194.2.9 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](#)

## 5.195 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](#)

### 5.195.1 Detailed Description

Phy type information.

Definition at line 143 of file [vtss\\_phy\\_api.h](#).

### 5.195.2 Field Documentation

#### 5.195.2.1 part\_number

```
u16 vtss_phy_type_t::part_number
```

Part number

Definition at line 145 of file [vtss\\_phy\\_api.h](#).

### 5.195.2.2 revision

`u16 vtss_phy_type_t::revision`

Chip revision

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

### 5.195.2.3 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`.

### 5.195.2.4 channel\_id

`u16 vtss_phy_type_t::channel_id`

Channel id

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

### 5.195.2.5 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`.

### 5.195.2.6 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`

## 5.196 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]`

#### 5.196.1 Detailed Description

VeriPHY result.

Definition at line 958 of file vtss\_phy\_api.h.

#### 5.196.2 Field Documentation

##### 5.196.2.1 link

```
BOOL vtss_phy_veriphy_result_t::link
```

Link status

Definition at line 959 of file vtss\_phy\_api.h.

##### 5.196.2.2 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.

### 5.196.2.3 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](#)

## 5.197 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`

### 5.197.1 Detailed Description

Structure for Get/Set Wake-On-LAN configuration.

Definition at line 1680 of file vtss\_phy\_api.h.

### 5.197.2 Field Documentation

#### 5.197.2.1 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.

### 5.197.2.2 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.

### 5.197.2.3 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.

### 5.197.2.4 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.

### 5.197.2.5 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](#)

## 5.198 vtss\_pi\_conf\_t Struct Reference

PI configuration.

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

## Data Fields

- `u32 cs_wait_ns`

### 5.198.1 Detailed Description

PI configuration.

Definition at line 322 of file `vtss_init_api.h`.

### 5.198.2 Field Documentation

#### 5.198.2.1 `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`

## 5.199 `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 known_unicast`
- `BOOL known_multicast`
- `BOOL known_broadcast`
- `BOOL unknown_unicast`
- `BOOL unknown_multicast`
- `BOOL unknown_broadcast`
- `BOOL learning`
- `BOOL to_cpu`
- `BOOL cpu_queue [VTSS_PORT_POLICER_CPU_QUEUES]`
- `BOOL limit_noncpu_traffic`
- `BOOL limit_cpu_traffic`
- `BOOL flow_control`

### 5.199.1 Detailed Description

Policer Extensions.

Definition at line 198 of file vtss\_qos\_api.h.

### 5.199.2 Field Documentation

#### 5.199.2.1 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.

#### 5.199.2.2 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.

#### 5.199.2.3 known\_unicast

`BOOL vtss_policer_ext_t::known_unicast`

Known unicast frames are policed

Definition at line 215 of file vtss\_qos\_api.h.

#### 5.199.2.4 known\_multicast

`BOOL vtss_policer_ext_t::known_multicast`

Known multicast frames are policed

Definition at line 216 of file vtss\_qos\_api.h.

### 5.199.2.5 known\_broadcast

`BOOL vtss_policer_ext_t::known_broadcast`

Known broadcast frames are policed

Definition at line 217 of file `vtss_qos_api.h`.

### 5.199.2.6 unknown\_unicast

`BOOL vtss_policer_ext_t::unknown_unicast`

Unknown unicast frames are policed

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

### 5.199.2.7 unknown\_multicast

`BOOL vtss_policer_ext_t::unknown_multicast`

Unknown multicast frames are policed

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

### 5.199.2.8 unknown\_broadcast

`BOOL vtss_policer_ext_t::unknown_broadcast`

Unknown broadcast frames are policed

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

### 5.199.2.9 learning

`BOOL vtss_policer_ext_t::learning`

Learning frames are policed

Definition at line 223 of file `vtss_qos_api.h`.

**5.199.2.10 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.

**5.199.2.11 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.

**5.199.2.12 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.

**5.199.2.13 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.

**5.199.2.14 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](#)

## 5.200 vtss\_policer\_t Struct Reference

Policer.

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

### Data Fields

- `vtss_burst_level_t level`
- `vtss_bitrate_t rate`

#### 5.200.1 Detailed Description

Policer.

Definition at line 185 of file `vtss_qos_api.h`.

#### 5.200.2 Field Documentation

##### 5.200.2.1 level

```
vtss_burst_level_t vtss_policer_t::level
```

Burst level

Definition at line 187 of file `vtss_qos_api.h`.

##### 5.200.2.2 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`

## 5.201 vtss\_port\_bridge\_counters\_t Struct Reference

Port bridge counter structure (RFC 4188)

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

## Data Fields

- [vtss\\_port\\_counter\\_t dot1dTpPortInDiscards](#)

### 5.201.1 Detailed Description

Port bridge counter structure (RFC 4188)

Definition at line 201 of file port.h.

### 5.201.2 Field Documentation

#### 5.201.2.1 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](#)

## 5.202 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](#)

### 5.202.1 Detailed Description

Advertisement control data for Clause 37 aneg.

Definition at line 127 of file vtss\_port\_api.h.

## 5.202.2 Field Documentation

### 5.202.2.1 fdx

`BOOL vtss_port_clause_37_adv_t::fdx`

(FD)

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

### 5.202.2.2 hdx

`BOOL vtss_port_clause_37_adv_t::hdx`

(HD)

Definition at line 130 of file vtss\_port\_api.h.

### 5.202.2.3 symmetric\_pause

`BOOL vtss_port_clause_37_adv_t::symmetric_pause`

(PS1)

Definition at line 131 of file vtss\_port\_api.h.

### 5.202.2.4 asymmetric\_pause

`BOOL vtss_port_clause_37_adv_t::asymmetric_pause`

(PS2)

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

### 5.202.2.5 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.

### 5.202.2.6 acknowledge

`BOOL vtss_port_clause_37_adv_t::acknowledge`

(Ack)

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

### 5.202.2.7 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`

## 5.203 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`

### 5.203.1 Detailed Description

Auto-negotiation control parameter struct.

Definition at line 152 of file vtss\_port\_api.h.

### 5.203.2 Field Documentation

### 5.203.2.1 enable

`BOOL vtss_port_clause_37_control_t::enable`

Enable of Autoneg

Definition at line 154 of file `vtss_port_api.h`.

### 5.203.2.2 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`

## 5.204 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`

### 5.204.1 Detailed Description

Port configuration structure.

Definition at line 268 of file vtss\_port\_api.h.

### 5.204.2 Field Documentation

#### 5.204.2.1 `if_type`

`vtss_port_interface_t` `vtss_port_conf_t::if_type`

Interface type

Definition at line 270 of file vtss\_port\_api.h.

#### 5.204.2.2 `sd_enable`

`BOOL` `vtss_port_conf_t::sd_enable`

Signal detect enable

Definition at line 271 of file vtss\_port\_api.h.

#### 5.204.2.3 `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.

#### 5.204.2.4 `sd_internal`

`BOOL` `vtss_port_conf_t::sd_internal`

Internal signal detect selection

Definition at line 273 of file vtss\_port\_api.h.

#### 5.204.2.5 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.

#### 5.204.2.6 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.

#### 5.204.2.7 speed

`vtss_port_speed_t` `vtss_port_conf_t::speed`

Port speed

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

#### 5.204.2.8 fdx

`BOOL` `vtss_port_conf_t::fdx`

Full duplex mode

Definition at line 277 of file vtss\_port\_api.h.

#### 5.204.2.9 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.

#### 5.204.2.10 max\_frame\_length

`u32 vtss_port_conf_t::max_frame_length`

Maximum frame length

Definition at line 279 of file vtss\_port\_api.h.

#### 5.204.2.11 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.

#### 5.204.2.12 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.

#### 5.204.2.13 exc\_col\_cont

`BOOL vtss_port_conf_t::exc_col_cont`

Excessive collision continuation

Definition at line 282 of file vtss\_port\_api.h.

#### 5.204.2.14 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.

### 5.204.2.15 xau\_tx\_lane\_flip

`BOOL vtss_port_conf_t::xau_tx_lane_flip`

Xau Tx lane flip

Definition at line 284 of file vtss\_port\_api.h.

### 5.204.2.16 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.

### 5.204.2.17 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`

## 5.205 vtss\_port\_counters\_t Struct Reference

Port counter structure.

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

### Data Fields

- `vtss_port_rmon_counters_t rmon`
- `vtss_port_if_group_counters_t if_group`
- `vtss_port_ethernet_like_counters_t ethernet_like`
- `vtss_port_bridge_counters_t bridge`
- `vtss_port_proprietary_counters_t prop`

### 5.205.1 Detailed Description

Port counter structure.

Definition at line 220 of file port.h.

### 5.205.2 Field Documentation

#### 5.205.2.1 rmon

```
vtss_port_rmon_counters_t vtss_port_counters_t::rmon
```

RMON counters

Definition at line 222 of file port.h.

#### 5.205.2.2 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.

#### 5.205.2.3 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.

#### 5.205.2.4 bridge

```
vtss_port_bridge_counters_t vtss_port_counters_t::bridge
```

Bridge counters

Definition at line 227 of file port.h.

### 5.205.2.5 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](#)

## 5.206 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](#)

### 5.206.1 Detailed Description

Ethernet-like Interface counter structure (RFC 3635)

Definition at line 163 of file port.h.

### 5.206.2 Field Documentation

### 5.206.2.1 dot3StatsAlignmentErrors

`vtss_port_counter_t vtss_port_ethernet_like_counters_t::dot3StatsAlignmentErrors`

Rx alignment errors

Definition at line 165 of file port.h.

### 5.206.2.2 dot3StatsFCSErrors

`vtss_port_counter_t vtss_port_ethernet_like_counters_t::dot3StatsFCSErrors`

Rx FCS errors

Definition at line 166 of file port.h.

### 5.206.2.3 dot3StatsFrameTooLongs

`vtss_port_counter_t vtss_port_ethernet_like_counters_t::dot3StatsFrameTooLongs`

Rx too long

Definition at line 167 of file port.h.

### 5.206.2.4 dot3StatsSymbolErrors

`vtss_port_counter_t vtss_port_ethernet_like_counters_t::dot3StatsSymbolErrors`

Rx symbol errors

Definition at line 168 of file port.h.

### 5.206.2.5 dot3ControlInUnknownOpcodes

`vtss_port_counter_t vtss_port_ethernet_like_counters_t::dot3ControlInUnknownOpcodes`

Rx unknown opcodes

Definition at line 169 of file port.h.

### 5.206.2.6 dot3InPauseFrames

`vtss_port_counter_t vtss_port_ethernet_like_counters_t::dot3InPauseFrames`

Rx pause

Definition at line 171 of file port.h.

### 5.206.2.7 dot3StatsSingleCollisionFrames

`vtss_port_counter_t vtss_port_ethernet_like_counters_t::dot3StatsSingleCollisionFrames`

Tx single collisions

Definition at line 174 of file port.h.

### 5.206.2.8 dot3StatsMultipleCollisionFrames

`vtss_port_counter_t vtss_port_ethernet_like_counters_t::dot3StatsMultipleCollisionFrames`

Tx multiple collisions

Definition at line 175 of file port.h.

### 5.206.2.9 dot3StatsDeferredTransmissions

`vtss_port_counter_t vtss_port_ethernet_like_counters_t::dot3StatsDeferredTransmissions`

Tx deferred

Definition at line 176 of file port.h.

### 5.206.2.10 dot3StatsLateCollisions

`vtss_port_counter_t vtss_port_ethernet_like_counters_t::dot3StatsLateCollisions`

Tx late collisions

Definition at line 177 of file port.h.

**5.206.2.11 dot3StatsExcessiveCollisions**

```
vtss_port_counter_t vtss_port_ethernet_like_counters_t::dot3StatsExcessiveCollisions
```

Tx excessive collisions

Definition at line 178 of file port.h.

**5.206.2.12 dot3StatsCarrierSenseErrors**

```
vtss_port_counter_t vtss_port_ethernet_like_counters_t::dot3StatsCarrierSenseErrors
```

Tx carrier sense errors

Definition at line 179 of file port.h.

**5.206.2.13 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

**5.207 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 \[VTSS\\_PRIOS\]](#)

### 5.207.1 Detailed Description

Flow control setup.

Definition at line 189 of file vtss\_port\_api.h.

### 5.207.2 Field Documentation

#### 5.207.2.1 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.

#### 5.207.2.2 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.

#### 5.207.2.3 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.

#### 5.207.2.4 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

## 5.208 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`

#### 5.208.1 Detailed Description

Inter frame gap structure.

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

#### 5.208.2 Field Documentation

##### 5.208.2.1 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.

##### 5.208.2.2 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.

### 5.208.2.3 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](#)

## 5.209 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](#)

### 5.209.1 Detailed Description

Interfaces Group counter structure (RFC 2863)

Definition at line 144 of file port.h.

### 5.209.2 Field Documentation

### 5.209.2.1 ifInOctets

`vtss_port_counter_t vtss_port_if_group_counters_t::ifInOctets`

Rx octets

Definition at line 145 of file port.h.

### 5.209.2.2 ifInUcastPkts

`vtss_port_counter_t vtss_port_if_group_counters_t::ifInUcastPkts`

Rx unicasts

Definition at line 146 of file port.h.

### 5.209.2.3 ifInMulticastPkts

`vtss_port_counter_t vtss_port_if_group_counters_t::ifInMulticastPkts`

Rx multicasts

Definition at line 147 of file port.h.

### 5.209.2.4 ifInBroadcastPkts

`vtss_port_counter_t vtss_port_if_group_counters_t::ifInBroadcastPkts`

Rx broadcasts

Definition at line 148 of file port.h.

### 5.209.2.5 ifInNUcastPkts

`vtss_port_counter_t vtss_port_if_group_counters_t::ifInNUcastPkts`

Rx non-unicasts

Definition at line 149 of file port.h.

### 5.209.2.6 ifInDiscards

```
vtss_port_counter_t vtss_port_if_group_counters_t::ifInDiscards
```

Rx discards

Definition at line 150 of file port.h.

### 5.209.2.7 ifInErrors

```
vtss_port_counter_t vtss_port_if_group_counters_t::ifInErrors
```

Rx errors

Definition at line 151 of file port.h.

### 5.209.2.8 ifOutOctets

```
vtss_port_counter_t vtss_port_if_group_counters_t::ifOutOctets
```

Tx octets

Definition at line 153 of file port.h.

### 5.209.2.9 ifOutUcastPkts

```
vtss_port_counter_t vtss_port_if_group_counters_t::ifOutUcastPkts
```

Tx unicasts

Definition at line 154 of file port.h.

### 5.209.2.10 ifOutMulticastPkts

```
vtss_port_counter_t vtss_port_if_group_counters_t::ifOutMulticastPkts
```

Tx multicasts

Definition at line 155 of file port.h.

**5.209.2.11 ifOutBroadcastPkts**

```
vtss_port_counter_t vtss_port_if_group_counters_t::ifOutBroadcastPkts
```

Tx broadcasts

Definition at line 156 of file port.h.

**5.209.2.12 ifOutNUcastPkts**

```
vtss_port_counter_t vtss_port_if_group_counters_t::ifOutNUcastPkts
```

Tx non-unicasts

Definition at line 157 of file port.h.

**5.209.2.13 ifOutDiscards**

```
vtss_port_counter_t vtss_port_if_group_counters_t::ifOutDiscards
```

Tx discards

Definition at line 158 of file port.h.

**5.209.2.14 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](#)

---

**5.210 vtss\_port\_ifh\_t Struct Reference**

Port Internal Frame Header structure.

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

## Data Fields

- `BOOL ena_inj_header`
- `BOOL ena_xtr_header`
- `BOOL ena_ifh_header`

### 5.210.1 Detailed Description

Port Internal Frame Header structure.

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

### 5.210.2 Field Documentation

#### 5.210.2.1 `ena_inj_header`

`BOOL vtss_port_ifh_t::ena_inj_header`

At ingress expect short prefix: DMAC:SMAC:0x8880:0007 followed by an internal frame header and then the frame

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

#### 5.210.2.2 `ena_xtr_header`

`BOOL vtss_port_ifh_t::ena_xtr_header`

At egress prepend short prefix: DMAC:SMAC:0x8880:0007 (DMAC:SMAC from frame) followed by an internal frame header and then the frame

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

#### 5.210.2.3 `ena_ifh_header`

`BOOL vtss_port_ifh_t::ena_ifh_header`

Same as `ena_xtr_header` (for compatibility with JR1)

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

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

- `vtss_api/include/vtss_port_api.h`

## 5.211 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_internal_bw_t max_bw`
- `vtss_miim_controller_t miim_controller`
- `u8 miim_addr`
- `vtss_chip_no_t miim_chip_no`

#### 5.211.1 Detailed Description

Port map structure.

Definition at line 72 of file vtss\_port\_api.h.

#### 5.211.2 Field Documentation

##### 5.211.2.1 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.

##### 5.211.2.2 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.

### 5.211.2.3 max\_bw

`vtss_internal_bw_t vtss_port_map_t::max_bw`

Max internal bandwidth reserved for the port

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

### 5.211.2.4 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`.

### 5.211.2.5 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`.

### 5.211.2.6 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`

## 5.212 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]

### 5.212.1 Detailed Description

Port proprietary counter structure.

Definition at line 207 of file port.h.

### 5.212.2 Field Documentation

#### 5.212.2.1 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.

#### 5.212.2.2 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`

## 5.213 `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`

### 5.213.1 Detailed Description

RMON counter structure (RFC 2819)

Definition at line 109 of file port.h.

### 5.213.2 Field Documentation

#### 5.213.2.1 `rx_etherStatsDropEvents`

```
vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsDropEvents
```

Rx drop events

Definition at line 110 of file port.h.

### 5.213.2.2 rx\_etherStatsOctets

`vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsOctets`

Rx octets

Definition at line 111 of file port.h.

### 5.213.2.3 rx\_etherStatsPkts

`vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsPkts`

Rx packets

Definition at line 112 of file port.h.

### 5.213.2.4 rx\_etherStatsBroadcastPkts

`vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsBroadcastPkts`

Rx broadcasts

Definition at line 113 of file port.h.

### 5.213.2.5 rx\_etherStatsMulticastPkts

`vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsMulticastPkts`

Rx multicasts

Definition at line 114 of file port.h.

### 5.213.2.6 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.

### 5.213.2.7 rx\_etherStatsUndersizePkts

`vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsUndersizePkts`

Rx undersize packets

Definition at line 116 of file port.h.

### 5.213.2.8 rx\_etherStatsOversizePkts

`vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsOversizePkts`

Rx oversize packets

Definition at line 117 of file port.h.

### 5.213.2.9 rx\_etherStatsFragments

`vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsFragments`

Rx fragments

Definition at line 118 of file port.h.

### 5.213.2.10 rx\_etherStatsJabbers

`vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsJabbers`

Rx jabbers

Definition at line 119 of file port.h.

### 5.213.2.11 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.

**5.213.2.12 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.

**5.213.2.13 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.

**5.213.2.14 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.

**5.213.2.15 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.

**5.213.2.16 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.

**5.213.2.17 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.

**5.213.2.18 tx\_etherStatsDropEvents**

```
vtss_port_counter_t vtss_port_rmon_counters_t::tx_etherStatsDropEvents
```

Tx drop events

Definition at line 128 of file port.h.

**5.213.2.19 tx\_etherStatsOctets**

```
vtss_port_counter_t vtss_port_rmon_counters_t::tx_etherStatsOctets
```

Tx octets

Definition at line 129 of file port.h.

**5.213.2.20 tx\_etherStatsPkts**

```
vtss_port_counter_t vtss_port_rmon_counters_t::tx_etherStatsPkts
```

Tx packets

Definition at line 130 of file port.h.

**5.213.2.21 tx\_etherStatsBroadcastPkts**

```
vtss_port_counter_t vtss_port_rmon_counters_t::tx_etherStatsBroadcastPkts
```

Tx broadcasts

Definition at line 131 of file port.h.

**5.213.2.22 tx\_etherStatsMulticastPkts**

```
vtss_port_counter_t vtss_port_rmon_counters_t::tx_etherStatsMulticastPkts
```

Tx multicasts

Definition at line 132 of file port.h.

**5.213.2.23 tx\_etherStatsCollisions**

```
vtss_port_counter_t vtss_port_rmon_counters_t::tx_etherStatsCollisions
```

Tx collisions

Definition at line 133 of file port.h.

**5.213.2.24 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.

**5.213.2.25 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.

**5.213.2.26 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.

**5.213.2.27 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.

**5.213.2.28 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.

**5.213.2.29 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.

**5.213.2.30 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](#)

## 5.214 vtss\_port\_serdes\_conf\_t Struct Reference

SFI Serdes configuration.

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

## Data Fields

- `BOOL sfp_dac`

### 5.214.1 Detailed Description

SFI Serdes configuration.

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

### 5.214.2 Field Documentation

#### 5.214.2.1 `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`

## 5.215 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`

### 5.215.1 Detailed Description

Advertisement control data for SGMII aneg.

Definition at line 139 of file vtss\_port\_api.h.

## 5.215.2 Field Documentation

### 5.215.2.1 link

`BOOL vtss_port_sgmii_aneg_t::link`

LP link status

Definition at line 141 of file `vtss_port_api.h`.

### 5.215.2.2 fdx

`BOOL vtss_port_sgmii_aneg_t::fdx`

FD

Definition at line 142 of file `vtss_port_api.h`.

### 5.215.2.3 hdx

`BOOL vtss_port_sgmii_aneg_t::hdx`

HD

Definition at line 143 of file `vtss_port_api.h`.

### 5.215.2.4 speed\_10M

`BOOL vtss_port_sgmii_aneg_t::speed_10M`

speed 10 advertised

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

### 5.215.2.5 speed\_100M

`BOOL vtss_port_sgmii_aneg_t::speed_100M`

speed 100 advertised

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

### 5.215.2.6 speed\_1G

`BOOL vtss_port_sgmii_aneg_t::speed_1G`

speed 1G advertised

Definition at line 146 of file vtss\_port\_api.h.

### 5.215.2.7 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`

## 5.216 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`

### 5.216.1 Detailed Description

Port status parameter struct.

Definition at line 295 of file port.h.

## 5.216.2 Field Documentation

### 5.216.2.1 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.

### 5.216.2.2 link

`BOOL vtss_port_status_t::link`

Link is up. Remaining fields only valid if TRUE

Definition at line 298 of file port.h.

### 5.216.2.3 speed

`vtss_port_speed_t vtss_port_status_t::speed`

Speed

Definition at line 299 of file port.h.

### 5.216.2.4 fdx

`BOOL vtss_port_status_t::fdx`

Full duplex

Definition at line 300 of file port.h.

### 5.216.2.5 remote\_fault

`BOOL vtss_port_status_t::remote_fault`

Remote fault signalled

Definition at line 301 of file port.h.

### 5.216.2.6 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.

### 5.216.2.7 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.

### 5.216.2.8 aneg

`vtss_aneg_t vtss_port_status_t::aneg`

Auto negotiation result

Definition at line 307 of file port.h.

### 5.216.2.9 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.

### 5.216.2.10 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.

### 5.216.2.11 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`

## 5.217 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`
- `BOOL pcp_dei_enable`
- `vtss_tagprio_t pcp`
- `vtss_dei_t dei`
- `BOOL policy_no_enable`
- `vtss_acl_policy_no_t policy_no`

### 5.217.1 Detailed Description

QCE action.

Definition at line 566 of file vtss\_qos\_api.h.

### 5.217.2 Field Documentation

#### 5.217.2.1 prio\_enable

`BOOL vtss_qce_action_t::prio_enable`

Enable priority classification

Definition at line 568 of file vtss\_qos\_api.h.

### 5.217.2.2 prio

`vtss_prio_t vtss_qce_action_t::prio`

Priority value

Definition at line 569 of file vtss\_qos\_api.h.

### 5.217.2.3 dp\_enable

`BOOL vtss_qce_action_t::dp_enable`

Enable DP classification

Definition at line 570 of file vtss\_qos\_api.h.

### 5.217.2.4 dp

`vtss_dp_level_t vtss_qce_action_t::dp`

DP value

Definition at line 571 of file vtss\_qos\_api.h.

### 5.217.2.5 dscp\_enable

`BOOL vtss_qce_action_t::dscp_enable`

Enable DSCP classification

Definition at line 572 of file vtss\_qos\_api.h.

### 5.217.2.6 dscp

`vtss_dscp_t vtss_qce_action_t::dscp`

DSCP value

Definition at line 573 of file vtss\_qos\_api.h.

### 5.217.2.7 pcp\_dei\_enable

`BOOL vtss_qce_action_t::pcp_dei_enable`

Enable PCP and DEI classification

Definition at line 575 of file `vtss_qos_api.h`.

### 5.217.2.8 pcp

`vtss_tagprio_t vtss_qce_action_t::pcp`

PCP value

Definition at line 576 of file `vtss_qos_api.h`.

### 5.217.2.9 dei

`vtss_dei_t vtss_qce_action_t::dei`

DEI value

Definition at line 577 of file `vtss_qos_api.h`.

### 5.217.2.10 policy\_no\_enable

`BOOL vtss_qce_action_t::policy_no_enable`

Enable ACL policy classification

Definition at line 580 of file `vtss_qos_api.h`.

### 5.217.2.11 policy\_no

`vtss_acl_policy_no_t vtss_qce_action_t::policy_no`

ACL policy number

Definition at line 581 of file `vtss_qos_api.h`.

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

- `vtss_api/include/vtss_qos_api.h`

## 5.218 vtss\_qce\_frame\_etype\_t Struct Reference

Frame data for VTSS\_QCE\_TYPEETYPE.

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

### Data Fields

- [vtss\\_vcap\\_u16\\_t etype](#)
- [vtss\\_vcap\\_u32\\_t data](#)

#### 5.218.1 Detailed Description

Frame data for VTSS\_QCE\_TYPEETYPE.

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

#### 5.218.2 Field Documentation

##### 5.218.2.1 etype

```
vtss_vcap_u16_t vtss_qce_frame_etype_t::etype
```

Ethernet Type value

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

##### 5.218.2.2 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](#)

## 5.219 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_ip_t` dip
- `vtss_vcap_vr_t` sport
- `vtss_vcap_vr_t` dport

### 5.219.1 Detailed Description

Frame data for VTSS\_QCE\_TYPE\_IPV4.

Definition at line 513 of file vtss\_qos\_api.h.

### 5.219.2 Field Documentation

#### 5.219.2.1 fragment

`vtss_vcap_bit_t` vtss\_qce\_frame\_ipv4\_t::fragment

Fragment

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

#### 5.219.2.2 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.

#### 5.219.2.3 proto

`vtss_vcap_u8_t` vtss\_qce\_frame\_ipv4\_t::proto

Protocol

Definition at line 517 of file vtss\_qos\_api.h.

#### 5.219.2.4 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.

#### 5.219.2.5 dip

`vtss_vcap_ip_t` `vtss_qce_frame_ipv4_t::dip`

Destination IP address - Serval: key\_type = ip\_addr and mac\_ip\_addr

Definition at line 520 of file vtss\_qos\_api.h.

#### 5.219.2.6 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.

#### 5.219.2.7 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](#)

## 5.220 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_u128_t dip`
- `vtss_vcap_vr_t sport`
- `vtss_vcap_vr_t dport`

### 5.220.1 Detailed Description

Frame data for VTSS\_QCE\_TYPE\_IPV6.

Definition at line 527 of file vtss\_qos\_api.h.

### 5.220.2 Field Documentation

#### 5.220.2.1 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.

#### 5.220.2.2 proto

`vtss_vcap_u8_t vtss_qce_frame_ipv6_t::proto`

Protocol

Definition at line 530 of file vtss\_qos\_api.h.

#### 5.220.2.3 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.

#### 5.220.2.4 dip

`vtss_vcap_u128_t vtss_qce_frame_ipv6_t::dip`

Destination IP address - 64 LSB on Serval when key\_type = mac\_ip\_addr

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

#### 5.220.2.5 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.

#### 5.220.2.6 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`

## 5.221 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`

#### 5.221.1 Detailed Description

Frame data for VTSS\_QCE\_TYPE\_LLC.

Definition at line 501 of file vtss\_qos\_api.h.

## 5.221.2 Field Documentation

### 5.221.2.1 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`

## 5.222 `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`

### 5.222.1 Detailed Description

Frame data for `VTSS_QCE_TYPE_SNAP`.

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

### 5.222.2 Field Documentation

#### 5.222.2.1 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`

## 5.223 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_tag_t inner_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`}

### 5.223.1 Detailed Description

QCE key.

Definition at line 542 of file vtss\_qos\_api.h.

### 5.223.2 Field Documentation

#### 5.223.2.1 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.

#### 5.223.2.2 mac

```
vtss_qce_mac_t vtss_qce_key_t::mac
```

MAC

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

### 5.223.2.3 tag

`vtss_qce_tag_t` `vtss_qce_key_t::tag`

Tag

Definition at line 546 of file vtss\_qos\_api.h.

### 5.223.2.4 inner\_tag

`vtss_qce_tag_t` `vtss_qce_key_t::inner_tag`

Inner tag

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

### 5.223.2.5 type

`vtss_qce_type_t` `vtss_qce_key_t::type`

Frame type

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

### 5.223.2.6 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.

### 5.223.2.7 llc

`vtss_qce_frame_llc_t` `vtss_qce_key_t::llc`

VTSS\_QCE\_TYPE\_LLCC

Definition at line 556 of file vtss\_qos\_api.h.

**5.223.2.8 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`.

**5.223.2.9 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`.

**5.223.2.10 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`.

**5.223.2.11 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`

---

**5.224 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 dmac`
- `vtss_vcap_u48_t smac`

### 5.224.1 Detailed Description

QCE MAC information.

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

### 5.224.2 Field Documentation

#### 5.224.2.1 dmac\_mc

`vtss_vcap_bit_t vtss_qce_mac_t::dmac_mc`

Multicast DMAC

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

#### 5.224.2.2 dmac\_bc

`vtss_vcap_bit_t vtss_qce_mac_t::dmac_bc`

Broadcast DMAC

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

#### 5.224.2.3 dmac

`vtss_vcap_u48_t vtss_qce_mac_t::dmac`

DMAC - Serval: key\_type = mac\_ip\_addr

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

#### 5.224.2.4 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`

## 5.225 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`

### 5.225.1 Detailed Description

QoS Control Entry.

Definition at line 588 of file `vtss_qos_api.h`.

### 5.225.2 Field Documentation

#### 5.225.2.1 id

`vtss_qce_id_t vtss_qce_t::id`

Entry ID

Definition at line 590 of file `vtss_qos_api.h`.

### 5.225.2.2 key

`vtss_qce_key_t` `vtss_qce_t::key`

QCE key

Definition at line 591 of file vtss\_qos\_api.h.

### 5.225.2.3 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](#)

## 5.226 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`
- `vtss_vcap_bit_t s_tag`

### 5.226.1 Detailed Description

QCE tag information.

Definition at line 482 of file vtss\_qos\_api.h.

### 5.226.2 Field Documentation

### 5.226.2.1 vid

`vtss_vcap_vr_t` `vtss_qce_tag_t::vid`

VLAN ID (12 bit)

Definition at line 484 of file `vtss_qos_api.h`.

### 5.226.2.2 pcp

`vtss_vcap_u8_t` `vtss_qce_tag_t::pcp`

PCP (3 bit)

Definition at line 485 of file `vtss_qos_api.h`.

### 5.226.2.3 dei

`vtss_vcap_bit_t` `vtss_qce_tag_t::dei`

DEI

Definition at line 486 of file `vtss_qos_api.h`.

### 5.226.2.4 tagged

`vtss_vcap_bit_t` `vtss_qce_tag_t::tagged`

Tagged/untagged frame

Definition at line 487 of file `vtss_qos_api.h`.

### 5.226.2.5 s\_tag

`vtss_vcap_bit_t` `vtss_qce_tag_t::s_tag`

S-tagged/C-tagged frame

Definition at line 489 of file `vtss_qos_api.h`.

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

- `vtss_api/include/vtss_qos_api.h`

## 5.227 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]`
- `vtss_dscp_t dscp_qos_map_dp1 [VTSS_PRIO_ARRAY_SIZE]`
- `vtss_dscp_t dscp_qos_map_dp2 [VTSS_PRIO_ARRAY_SIZE]`
- `vtss_dscp_t dscp_qos_map_dp3 [VTSS_PRIO_ARRAY_SIZE]`
- `BOOL dscp_remark [64]`
- `vtss_dscp_t dscp_translate_map [64]`
- `vtss_dscp_t dscp_remap [64]`
- `vtss_packet_rate_t policer_uc`
- `BOOL policer_uc_frame_rate`
- `vtss_storm_policer_mode_t policer_uc_mode`
- `vtss_packet_rate_t policer_mc`
- `BOOL policer_mc_frame_rate`
- `vtss_storm_policer_mode_t policer_mc_mode`
- `vtss_packet_rate_t policer_bc`
- `BOOL policer_bc_frame_rate`
- `vtss_storm_policer_mode_t policer_bc_mode`
- `vtss_red_v3_t red_v3 [VTSS_QUEUE_ARRAY_SIZE][VTSS_WRED_DPL_CNT][VTSS_WRED_GROUP_CNT]`

### 5.227.1 Detailed Description

All parameters below are defined per chip.

Definition at line 91 of file vtss\_qos\_api.h.

### 5.227.2 Field Documentation

#### 5.227.2.1 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.

### 5.227.2.2 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.

### 5.227.2.3 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.

### 5.227.2.4 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.

### 5.227.2.5 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.

### 5.227.2.6 dscp\_qos\_map\_dp1

```
vtss_dscp_t vtss_qos_conf_t::dscp_qos_map_dp1[VTSS_PRIO_ARRAY_SIZE]
```

Ingress: Mapping from QoS class to DSCP (DP aware and DP level = 1)

Definition at line 102 of file vtss\_qos\_api.h.

### 5.227.2.7 dscp\_qos\_map\_dp2

```
vtss_dscp_t vtss_qos_conf_t::dscp_qos_map_dp2[VTSS_PRIO_ARRAY_SIZE]
```

Ingress: Mapping from QoS class to DSCP (DP aware and DP level = 2)

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

### 5.227.2.8 dscp\_qos\_map\_dp3

```
vtss_dscp_t vtss_qos_conf_t::dscp_qos_map_dp3[VTSS_PRIO_ARRAY_SIZE]
```

Ingress: Mapping from QoS class to DSCP (DP aware and DP level = 3)

Definition at line 105 of file vtss\_qos\_api.h.

### 5.227.2.9 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.

### 5.227.2.10 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.

### 5.227.2.11 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.

### 5.227.2.12 policer\_uc

`vtss_packet_rate_t` vtss\_qos\_conf\_t::policer\_uc

Unicast packet storm policer

Definition at line 127 of file vtss\_qos\_api.h.

### 5.227.2.13 policer\_uc\_frame\_rate

`BOOL` vtss\_qos\_conf\_t::policer\_uc\_frame\_rate

FALSE: Unit is kbps. TRUE: Unit is fps

Definition at line 128 of file vtss\_qos\_api.h.

### 5.227.2.14 policer\_uc\_mode

`vtss_storm_policer_mode_t` vtss\_qos\_conf\_t::policer\_uc\_mode

Unicast packet storm policer mode

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

### 5.227.2.15 policer\_mc

`vtss_packet_rate_t` vtss\_qos\_conf\_t::policer\_mc

Multicast packet storm policer

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

### 5.227.2.16 policer\_mc\_frame\_rate

`BOOL` vtss\_qos\_conf\_t::policer\_mc\_frame\_rate

FALSE: Unit is kbps. TRUE: Unit is fps

Definition at line 133 of file vtss\_qos\_api.h.

### 5.227.2.17 policer\_mc\_mode

`vtss_storm_policer_mode_t` `vtss_qos_conf_t::policer_mc_mode`

Multicast packet storm policer mode

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

### 5.227.2.18 policer\_bc

`vtss_packet_rate_t` `vtss_qos_conf_t::policer_bc`

Broadcast packet storm policer

Definition at line 137 of file `vtss_qos_api.h`.

### 5.227.2.19 policer\_bc\_frame\_rate

`BOOL` `vtss_qos_conf_t::policer_bc_frame_rate`

FALSE: Unit is kbps. TRUE: Unit is fps

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

### 5.227.2.20 policer\_bc\_mode

`vtss_storm_policer_mode_t` `vtss_qos_conf_t::policer_bc_mode`

Broadcast packet storm policer mode

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

### 5.227.2.21 red\_v3

`vtss_red_v3_t` `vtss_qos_conf_t::red_v3[VTSS_QUEUE_ARRAY_SIZE][VTSS_WRED_DPL_CNT][VTSS_WRED_GROUP_CNT]`

< Random Early Detection - per queue (0..7), per dpl (1..3), per group (0..2)

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

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

- `vtss_api/include/vtss_qos_api.h`

## 5.228 vtss\_qos\_port\_conf\_t Struct Reference

QoS setup per port.

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

### Data Fields

- `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]`
- `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`
- `u8 dwrr_cnt`
- `vtss_pot_t queue_pct [VTSS_QUEUE_ARRAY_SIZE]`
- `vtss_wred_group_t wred_group`

### 5.228.1 Detailed Description

QoS setup per port.

Definition at line 344 of file vtss\_qos\_api.h.

### 5.228.2 Field Documentation

#### 5.228.2.1 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.

### 5.228.2.2 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.

### 5.228.2.3 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.

### 5.228.2.4 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.

### 5.228.2.5 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.

### 5.228.2.6 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.

### 5.228.2.7 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.

### 5.228.2.8 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.

### 5.228.2.9 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.

### 5.228.2.10 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.

### 5.228.2.11 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.

### 5.228.2.12 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.

### 5.228.2.13 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.

### 5.228.2.14 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.

### 5.228.2.15 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.

### 5.228.2.16 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.

**5.228.2.17 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.

**5.228.2.18 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.

**5.228.2.19 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.

**5.228.2.20 tag\_pcp\_map**

`vtss_tagprio_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.

**5.228.2.21 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.

### 5.228.2.22 dwrr\_enable

`BOOL vtss_qos_port_conf_t::dwrr_enable`

Enable Weighted fairness queueing

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

### 5.228.2.23 dwrr\_cnt

`u8 vtss_qos_port_conf_t::dwrr_cnt`

Number of queues, starting from queue 0, running in DWRR mode

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

### 5.228.2.24 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`.

### 5.228.2.25 wred\_group

`vtss_wred_group_t vtss_qos_port_conf_t::wred_group`

WRED group number - zero based

Definition at line 416 of file `vtss_qos_api.h`.

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

- `vtss_api/include/vtss_qos_api.h`

## 5.229 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`

### 5.229.1 Detailed Description

Structure for Get PHY RC-PLL status.

Definition at line 1742 of file `vtss_phy_api.h`.

### 5.229.2 Field Documentation

#### 5.229.2.1 `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`.

#### 5.229.2.2 `cal_error`

```
u8 vtss_rcpll_status_t::cal_error
```

Calibration Error indication

Definition at line 1744 of file `vtss_phy_api.h`.

#### 5.229.2.3 `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`

## 5.230 vtss\_red\_v2\_t Struct Reference

Random Early Detection configuration struct version 2 (per queue, per dpl - switch global)

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

### Data Fields

- `BOOL enable`
- `vtss_pct_t min_fl`
- `vtss_pct_t max`
- `vtss_wred_v2_max_t max_unit`

#### 5.230.1 Detailed Description

Random Early Detection configuration struct version 2 (per queue, per dpl - switch global)

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

#### 5.230.2 Field Documentation

##### 5.230.2.1 enable

```
BOOL vtss_red_v2_t::enable
```

Enable/disable RED

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

##### 5.230.2.2 min\_fl

```
vtss_pct_t vtss_red_v2_t::min_fl
```

Minimum fill level

Definition at line 76 of file `vtss_qos_api.h`.

### 5.230.2.3 max

`vtss_pct_t vtss_red_v2_t::max`

Maximum drop probability or fill level - selected by max\_unit

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

### 5.230.2.4 max\_unit

`vtss_wred_v2_max_t vtss_red_v2_t::max_unit`

Selects the unit for max

Definition at line 78 of file vtss\_qos\_api.h.

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

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

## 5.231 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](#)

### 5.231.1 Detailed Description

Restart status.

Definition at line 608 of file vtss\_init\_api.h.

### 5.231.2 Field Documentation

### 5.231.2.1 restart

```
vtss_restart_t vtss_restart_status_t::restart
```

Previous restart mode

Definition at line 609 of file vtss\_init\_api.h.

### 5.231.2.2 prev\_version

```
vtss_version_t vtss_restart_status_t::prev_version
```

Previous API version

Definition at line 610 of file vtss\_init\_api.h.

### 5.231.2.3 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](#)

## 5.232 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](#)

### 5.232.1 Detailed Description

Routing entry.

Definition at line 868 of file types.h.

### 5.232.2 Field Documentation

#### 5.232.2.1 type

`vtss_routing_entry_type_t vtss_routing_entry_t::type`

Type of route

Definition at line 871 of file types.h.

#### 5.232.2.2 ipv4\_uc

`vtss_ipv4_uc_t vtss_routing_entry_t::ipv4_uc`

IPv6 unicast route

Definition at line 875 of file types.h.

#### 5.232.2.3 ipv6\_uc

`vtss_ipv6_uc_t vtss_routing_entry_t::ipv6_uc`

IPv6 unicast route

Definition at line 878 of file types.h.

#### 5.232.2.4 route

`union { ... } vtss_routing_entry_t::route`

Route

### 5.232.2.5 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](#)

## 5.233 vtss\_secure\_on\_passwd\_t Struct Reference

Structure for Wake-On-LAN Secure-On Password.

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

### Data Fields

- [u8 passwd \[MAX\\_WOL\\_PASSWD\\_SIZE\]](#)

### 5.233.1 Detailed Description

Structure for Wake-On-LAN Secure-On Password.

Definition at line 1664 of file vtss\_phy\_api.h.

### 5.233.2 Field Documentation

#### 5.233.2.1 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](#)

## 5.234 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`

#### 5.234.1 Detailed Description

Serdes macro configuration.

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

#### 5.234.2 Field Documentation

##### 5.234.2.1 serdes1g\_vdd

```
vtss_vdd_t vtss_serdes_macro_conf_t::serdes1g_vdd
```

Serdes1g supply

Definition at line 364 of file `vtss_init_api.h`.

##### 5.234.2.2 serdes6g\_vdd

```
vtss_vdd_t vtss_serdes_macro_conf_t::serdes6g_vdd
```

Serdes6g supply

Definition at line 365 of file `vtss_init_api.h`.

### 5.234.2.3 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`.

### 5.234.2.4 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`

## 5.235 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`

### 5.235.1 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`.

### 5.235.2 Field Documentation

### 5.235.2.1 type

```
vtss_sf_low_type_t vtss_sf_low_port_conf_t::type
```

Sample direction. Also used to turn off sampling.

Definition at line 1451 of file vtss\_l2\_api.h.

### 5.235.2.2 sampling\_rate

```
u32 vtss_sf_low_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](#)

## 5.236 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]

### 5.236.1 Detailed Description

SGPIO configuration for a group.

Definition at line 797 of file vtss\_misc\_api.h.

### 5.236.2 Field Documentation

### 5.236.2.1 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`.

### 5.236.2.2 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`.

### 5.236.2.3 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`

## 5.237 vtss\_sgpio\_port\_conf\_t Struct Reference

SGPIO port configuration.

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

### Data Fields

- `BOOL enabled`
- `vtss_sgpio_mode_t mode [4]`
- `BOOL int_pol_high [4]`

### 5.237.1 Detailed Description

SGPIO port configuration.

Definition at line 789 of file `vtss_misc_api.h`.

## 5.237.2 Field Documentation

### 5.237.2.1 enabled

`BOOL vtss_sgpio_port_conf_t::enabled`

Port enabled/disabled

Definition at line 791 of file `vtss_misc_api.h`.

### 5.237.2.2 mode

`vtss_sgpio_mode_t vtss_sgpio_port_conf_t::mode[4]`

Mode for each bit

Definition at line 792 of file `vtss_misc_api.h`.

### 5.237.2.3 int\_pol\_high

`BOOL vtss_sgpio_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`

## 5.238 `vtss_sgpio_port_data_t` Struct Reference

SGPIO read data for a port.

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

### Data Fields

- `BOOL value [4]`

### 5.238.1 Detailed Description

SGPIO read data for a port.

Definition at line 835 of file vtss\_misc\_api.h.

### 5.238.2 Field Documentation

#### 5.238.2.1 value

```
BOOL vtss_sgpio_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](#)

## 5.239 vtss\_shaper\_t Struct Reference

Shaper.

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

### Data Fields

- [vtss\\_burst\\_level\\_t](#) level
- [vtss\\_bitrate\\_t](#) rate
- [vtss\\_burst\\_level\\_t](#) ebs
- [vtss\\_bitrate\\_t](#) eir

### 5.239.1 Detailed Description

Shaper.

Definition at line 298 of file vtss\_qos\_api.h.

### 5.239.2 Field Documentation

### 5.239.2.1 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`.

### 5.239.2.2 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`.

### 5.239.2.3 ebs

`vtss_burst_level_t` `vtss_shaper_t::ebs`

EBS (Excess Burst Size). Unit: bytes

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

### 5.239.2.4 eir

`vtss_bitrate_t` `vtss_shaper_t::eir`

EIR (Excess Information Rate). Unit: kbps. Use `VTSS_BITRATE_DISABLED` to disable DLB

Definition at line 304 of file `vtss_qos_api.h`.

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

- `vtss_api/include/vtss_qos_api.h`

## 5.240 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`

### 5.240.1 Detailed Description

10G Phy link and fault status

Definition at line 86 of file `vtss_phy_10g_api.h`.

### 5.240.2 Field Documentation

#### 5.240.2.1 rx\_link

`BOOL vtss_sublayer_status_t::rx_link`

The rx link status

Definition at line 87 of file `vtss_phy_10g_api.h`.

#### 5.240.2.2 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`.

#### 5.240.2.3 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`.

## 5.240.2.4 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`

## 5.241 **`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 squelch`
- `BOOL enable`

#### 5.241.1 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`.

#### 5.241.2 Field Documentation

##### 5.241.2.1 `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`.

### 5.241.2.2 squelsh

`BOOL vtss_sync_clock_in_t::squelsh`

Enable/disable of automatic squelsh

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

### 5.241.2.3 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`

## 5.242 `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`

### 5.242.1 Detailed Description

Struct containing configuration for a recovered clock output port.

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

### 5.242.2 Field Documentation

### 5.242.2.1 divider

```
vtss_sync_clock_out_t::divider
```

Selection the clock division. This should be set to VTSS\_SYNCE\_DIVIDER\_1 if recovered clock is comming from internal PHY

Definition at line 75 of file vtss\_sync\_api.h.

### 5.242.2.2 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](#)

## 5.243 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](#)

### 5.243.1 Detailed Description

Tag Control Information (according to IEEE 802.1Q)

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

### 5.243.2 Field Documentation

### 5.243.2.1 vid

`vtss_vid_t` `vtss_tci_t::vid`

VLAN ID

Definition at line 208 of file `vtss_packet_api.h`.

### 5.243.2.2 cfi

`BOOL` `vtss_tci_t::cfi`

Canonical Format Indicator

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

### 5.243.2.3 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`

## 5.244 vtss\_timeofday\_t Struct Reference

Time of day structure.

```
#include <vtss_os_ecos.h>
```

### Data Fields

- `u32 sec`
- `time_t sec`

### 5.244.1 Detailed Description

Time of day structure.

Definition at line 59 of file `vtss_os_ecos.h`.

## 5.244.2 Field Documentation

### 5.244.2.1 sec [1/2]

`u32 vtss_timeofday_t::sec`

Time of day in seconds

Definition at line 60 of file vtss\_os\_ecos.h.

### 5.244.2.2 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`

## 5.245 vtss\_timestamp\_t Struct Reference

Time stamp in seconds and nanoseconds.

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

### Data Fields

- `u16 sec_msb`
- `u32 seconds`
- `u32 nanoseconds`

### 5.245.1 Detailed Description

Time stamp in seconds and nanoseconds.

Definition at line 1212 of file types.h.

## 5.245.2 Field Documentation

### 5.245.2.1 sec\_msb

`u16 vtss_timestamp_t::sec_msb`

Seconds msb

Definition at line 1213 of file types.h.

### 5.245.2.2 seconds

`u32 vtss_timestamp_t::seconds`

Seconds

Definition at line 1214 of file types.h.

### 5.245.2.3 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`

## 5.246 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]`

### 5.246.1 Detailed Description

Trace group configuration.

Definition at line 97 of file vtss\_misc\_api.h.

### 5.246.2 Field Documentation

#### 5.246.2.1 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

## 5.247 vtss\_ts\_alt\_clock\_mode\_t Struct Reference

parameter for setting the alternative clock mode.

```
#include <vtss_ts_api.h>
```

### Data Fields

- `BOOL one_pps_out`
- `BOOL one_pps_in`
- `BOOL save`
- `BOOL load`

### 5.247.1 Detailed Description

parameter for setting the alternative clock mode.

external clock output configuration.

Definition at line 222 of file vtss\_ts\_api.h.

### 5.247.2 Field Documentation

### 5.247.2.1 one\_pps\_out

`BOOL vtss_ts_alt_clock_mode_t::one_pps_out`

Enable 1pps output

Definition at line 223 of file vtss\_ts\_api.h.

### 5.247.2.2 one\_pps\_in

`BOOL vtss_ts_alt_clock_mode_t::one_pps_in`

Enable 1pps input

Definition at line 224 of file vtss\_ts\_api.h.

### 5.247.2.3 save

`BOOL vtss_ts_alt_clock_mode_t::save`

Save actual time counter at next 1 PPS input

Definition at line 225 of file vtss\_ts\_api.h.

### 5.247.2.4 load

`BOOL vtss_ts_alt_clock_mode_t::load`

Load actual time counter with at next 1 PPS input

Definition at line 226 of file vtss\_ts\_api.h.

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

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

## 5.248 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`

### 5.248.1 Detailed Description

external clock output configuration.

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

### 5.248.2 Field Documentation

#### 5.248.2.1 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`.

#### 5.248.2.2 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`.

#### 5.248.2.3 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`

## 5.249 vtss\_ts\_id\_t Struct Reference

Timestamp identifier.

```
#include <vtss_ts_api.h>
```

### Data Fields

- [u32 ts\\_id](#)

#### 5.249.1 Detailed Description

Timestamp identifier.

Definition at line 527 of file vtss\_ts\_api.h.

#### 5.249.2 Field Documentation

##### 5.249.2.1 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](#)

## 5.250 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](#)

### 5.250.1 Detailed Description

Hardware timestamping format mode for internal ports.

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

### 5.250.2 Field Documentation

#### 5.250.2.1 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](#)

## 5.251 vtss\_ts\_operation\_mode\_t Struct Reference

Timestamp operation.

```
#include <vtss_ts_api.h>
```

### Data Fields

- [vtss\\_ts\\_mode\\_t mode](#)

### 5.251.1 Detailed Description

Timestamp operation.

Definition at line 451 of file vtss\_ts\_api.h.

### 5.251.2 Field Documentation

### 5.251.2.1 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`

## 5.252 `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)`

### 5.252.1 Detailed Description

Timestamp allocation.

Definition at line 613 of file `vtss_ts_api.h`.

### 5.252.2 Field Documentation

#### 5.252.2.1 `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`.

### 5.252.2.2 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.

### 5.252.2.3 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](#)

## 5.253 vtss\_ts\_timestamp\_t Struct Reference

Timestamp structure.

```
#include <vtss_ts_api.h>
```

### Data Fields

- [u32 ts](#)
- [u32 id](#)
- [void \\* context](#)
- [BOOL ts\\_valid](#)

### 5.253.1 Detailed Description

Timestamp structure.

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

### 5.253.2 Field Documentation

### 5.253.2.1 ts

`u32 vtss_ts_timestamp_t::ts`

Timestamp value

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

### 5.253.2.2 id

`u32 vtss_ts_timestamp_t::id`

Timestamp identifier

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

### 5.253.2.3 context

`void* vtss_ts_timestamp_t::context`

Application specific context

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

### 5.253.2.4 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`

## 5.254 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

### 5.254.1 Detailed Description

VCAP IPv4 address value and mask.

Definition at line 968 of file types.h.

### 5.254.2 Field Documentation

#### 5.254.2.1 value

`vtss_ip_t` vtss\_vcap\_ip\_t::value

Value

Definition at line 970 of file types.h.

#### 5.254.2.2 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`

## 5.255 vtss\_vcap\_u128\_t Struct Reference

VCAP 128 bit value and mask.

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

## Data Fields

- `u8` value [16]
- `u8` mask [16]

### 5.255.1 Detailed Description

VCAP 128 bit value and mask.

Definition at line 954 of file types.h.

### 5.255.2 Field Documentation

#### 5.255.2.1 value

```
u8 vtss_vcap_u128_t::value[16]
```

Value

Definition at line 956 of file types.h.

#### 5.255.2.2 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](#)

## 5.256 vtss\_vcap\_u16\_t Struct Reference

VCAP 16 bit value and mask.

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

### Data Fields

- [u8 value \[2\]](#)
- [u8 mask \[2\]](#)

### 5.256.1 Detailed Description

VCAP 16 bit value and mask.

Definition at line 919 of file types.h.

### 5.256.2 Field Documentation

#### 5.256.2.1 value

`u8 vtss_vcap_u16_t::value[2]`

Value

Definition at line 921 of file types.h.

#### 5.256.2.2 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`

## 5.257 vtss\_vcap\_u24\_t Struct Reference

VCAP 24 bit value and mask.

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

### Data Fields

- `u8 value [3]`
- `u8 mask [3]`

### 5.257.1 Detailed Description

VCAP 24 bit value and mask.

Definition at line 926 of file types.h.

### 5.257.2 Field Documentation

#### 5.257.2.1 value

`u8 vtss_vcap_u24_t::value[3]`

Value

Definition at line 928 of file types.h.

#### 5.257.2.2 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`

## 5.258 vtss\_vcap\_u32\_t Struct Reference

VCAP 32 bit value and mask.

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

### Data Fields

- `u8 value [4]`
- `u8 mask [4]`

### 5.258.1 Detailed Description

VCAP 32 bit value and mask.

Definition at line 933 of file types.h.

### 5.258.2 Field Documentation

#### 5.258.2.1 value

`u8 vtss_vcap_u32_t::value[4]`

Value

Definition at line 935 of file types.h.

#### 5.258.2.2 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`

## 5.259 vtss\_vcap\_u40\_t Struct Reference

VCAP 40 bit value and mask.

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

### Data Fields

- `u8 value [5]`
- `u8 mask [5]`

### 5.259.1 Detailed Description

VCAP 40 bit value and mask.

Definition at line 940 of file types.h.

### 5.259.2 Field Documentation

#### 5.259.2.1 value

```
u8 vtss_vcap_u40_t::value[5]
```

Value

Definition at line 942 of file types.h.

#### 5.259.2.2 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](#)

## 5.260 vtss\_vcap\_u48\_t Struct Reference

VCAP 48 bit value and mask.

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

### Data Fields

- [u8 value \[6\]](#)
- [u8 mask \[6\]](#)

### 5.260.1 Detailed Description

VCAP 48 bit value and mask.

Definition at line 947 of file types.h.

### 5.260.2 Field Documentation

#### 5.260.2.1 value

`u8 vtss_vcap_u48_t::value[6]`

Value

Definition at line 949 of file types.h.

#### 5.260.2.2 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`

## 5.261 vtss\_vcap\_u8\_t Struct Reference

VCAP 8 bit value and mask.

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

### Data Fields

- `u8 value`
- `u8 mask`

### 5.261.1 Detailed Description

VCAP 8 bit value and mask.

Definition at line 912 of file types.h.

### 5.261.2 Field Documentation

#### 5.261.2.1 value

```
u8 vtss_vcap_u8_t::value
```

Value

Definition at line 914 of file types.h.

#### 5.261.2.2 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](#)

## 5.262 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](#)

### 5.262.1 Detailed Description

VCAP UDP/TCP port range.

Definition at line 975 of file types.h.

### 5.262.2 Field Documentation

#### 5.262.2.1 in\_range

`BOOL vtss_vcap_udp_tcp_t::in_range`

Port in range match

Definition at line 977 of file types.h.

#### 5.262.2.2 low

`vtss_udp_tcp_t vtss_vcap_udp_tcp_t::low`

Port low value

Definition at line 978 of file types.h.

#### 5.262.2.3 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`

## 5.263 vtss\_vcap\_vid\_t Struct Reference

VCAP VLAN ID value and mask.

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

## Data Fields

- [u16 value](#)
- [u16 mask](#)

### 5.263.1 Detailed Description

VCAP VLAN ID value and mask.

Definition at line 961 of file `types.h`.

### 5.263.2 Field Documentation

#### 5.263.2.1 value

`u16 vtss_vcap_vid_t::value`

Value

Definition at line 963 of file `types.h`.

#### 5.263.2.2 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`

## 5.264 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
```

### 5.264.1 Detailed Description

VCAP universal value or range.

Definition at line 994 of file types.h.

### 5.264.2 Field Documentation

#### 5.264.2.1 type

```
vtss_vcap_vr_type_t vtss_vcap_vr_t::type
```

Type

Definition at line 996 of file types.h.

#### 5.264.2.2 value

```
vtss_vcap_vr_value_t vtss_vcap_vr_t::value
```

Value

Definition at line 1001 of file types.h.

### 5.264.2.3 mask

```
vtss_vcap_vr_value_t vtss_vcap_vr_t::mask
```

Mask, cleared bits are wildcards

Definition at line 1002 of file types.h.

### 5.264.2.4 v

```
struct { ... } vtss_vcap_vr_t::v  
type == VTSS_VCAP_VR_TYPE_VALUE_MASK
```

### 5.264.2.5 low

```
vtss_vcap_vr_value_t vtss_vcap_vr_t::low
```

Low value

Definition at line 1006 of file types.h.

### 5.264.2.6 high

```
vtss_vcap_vr_value_t vtss_vcap_vr_t::high
```

High value

Definition at line 1007 of file types.h.

### 5.264.2.7 r

```
struct { ... } vtss_vcap_vr_t::r  
type == VTSS_VCAP_VR_TYPE_RANGE_XXXXXX
```

### 5.264.2.8 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](#)

## 5.265 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](#)

#### 5.265.1 Detailed Description

VCE Action.

Definition at line 1006 of file vtss\_l2\_api.h.

#### 5.265.2 Field Documentation

##### 5.265.2.1 vid

```
vtss_vid_t vtss_vce_action_t::vid
```

Classified VLAN ID

Definition at line 1008 of file vtss\_l2\_api.h.

##### 5.265.2.2 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](#)

## 5.266 vtss\_vce\_frame\_etype\_t Struct Reference

Frame data for VTSS\_VCE\_TYPEETYPE.

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

## Data Fields

- `vtss_vcap_u16_t` `etype`
- `vtss_vcap_u32_t` `data`

### 5.266.1 Detailed Description

Frame data for VTSS\_VCE\_TYPEETYPE.

Definition at line 948 of file vtss\_l2\_api.h.

### 5.266.2 Field Documentation

#### 5.266.2.1 `etype`

`vtss_vcap_u16_t` `vtss_vce_frame_etype_t::etype`

Ethernet Type value

Definition at line 950 of file vtss\_l2\_api.h.

#### 5.266.2.2 `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](#)

## 5.267 `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

### 5.267.1 Detailed Description

Frame data for VTSS\_VCE\_TYPE\_IPV4.

Definition at line 967 of file vtss\_l2\_api.h.

### 5.267.2 Field Documentation

#### 5.267.2.1 fragment

`vtss_vcap_bit_t` vtss\_vce\_frame\_ipv4\_t::fragment

Fragment

Definition at line 969 of file vtss\_l2\_api.h.

#### 5.267.2.2 options

`vtss_vcap_bit_t` vtss\_vce\_frame\_ipv4\_t::options

Header options

Definition at line 970 of file vtss\_l2\_api.h.

#### 5.267.2.3 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.

#### 5.267.2.4 proto

`vtss_vcap_u8_t vtss_vce_frame_ipv4_t::proto`

Protocol

Definition at line 972 of file `vtss_l2_api.h`.

#### 5.267.2.5 sip

`vtss_vcap_ip_t vtss_vce_frame_ipv4_t::sip`

Source IP address

Definition at line 973 of file `vtss_l2_api.h`.

#### 5.267.2.6 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`

## 5.268 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`

### 5.268.1 Detailed Description

Frame data for VTSS\_VCE\_TYPE\_IPV6.

Definition at line 978 of file vtss\_l2\_api.h.

### 5.268.2 Field Documentation

#### 5.268.2.1 dscp

`vtss_vcap_vr_t` `vtss_vce_frame_ipv6_t::dscp`

DSCH field (6 bit)

Definition at line 980 of file vtss\_l2\_api.h.

#### 5.268.2.2 proto

`vtss_vcap_u8_t` `vtss_vce_frame_ipv6_t::proto`

Protocol

Definition at line 981 of file vtss\_l2\_api.h.

#### 5.268.2.3 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.

#### 5.268.2.4 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](#)

## 5.269 vtss\_vce\_frame\_llc\_t Struct Reference

Frame data for VTSS\_VCE\_TYPE\_LLCC.

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

### Data Fields

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

#### 5.269.1 Detailed Description

Frame data for VTSS\_VCE\_TYPE\_LLCC.

Definition at line 955 of file vtss\_l2\_api.h.

#### 5.269.2 Field Documentation

##### 5.269.2.1 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](#)

## 5.270 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](#)

### 5.270.1 Detailed Description

Frame data for VTSS\_VCE\_TYPE\_SNAP.

Definition at line 961 of file vtss\_l2\_api.h.

### 5.270.2 Field Documentation

#### 5.270.2.1 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`

## 5.271 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``}` `frame`

### 5.271.1 Detailed Description

VCE Key.

Definition at line 987 of file vtss\_l2\_api.h.

## 5.271.2 Field Documentation

### 5.271.2.1 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.

### 5.271.2.2 mac

```
vtss_vce_mac_t vtss_vce_key_t::mac
```

MAC header

Definition at line 990 of file vtss\_l2\_api.h.

### 5.271.2.3 tag

```
vtss_vce_tag_t vtss_vce_key_t::tag
```

Tag

Definition at line 991 of file vtss\_l2\_api.h.

### 5.271.2.4 type

```
vtss_vce_type_t vtss_vce_key_t::type
```

VCE frame type

Definition at line 992 of file vtss\_l2\_api.h.

### 5.271.2.5 etype

```
vtss_vce_frame_etype_t vtss_vce_key_t::etype
```

VTSS\_VCE\_TYPEETYPE

Definition at line 997 of file vtss\_l2\_api.h.

### 5.271.2.6 llc

`vtss_vce_frame_llc_t vtss_vce_key_t::llc`

VTSS\_VCE\_TYPE LLC

Definition at line 998 of file vtss\_l2\_api.h.

### 5.271.2.7 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.

### 5.271.2.8 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.

### 5.271.2.9 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.

### 5.271.2.10 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](#)

## 5.272 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`

#### 5.272.1 Detailed Description

VCE MAC header information.

Definition at line 930 of file vtss\_l2\_api.h.

#### 5.272.2 Field Documentation

##### 5.272.2.1 dmac\_mc

```
vtss_vcap_bit_t vtss_vce_mac_t::dmac_mc
```

Multicast DMAc

Definition at line 932 of file vtss\_l2\_api.h.

##### 5.272.2.2 dmac\_bc

```
vtss_vcap_bit_t vtss_vce_mac_t::dmac_bc
```

Broadcast DMAc

Definition at line 933 of file vtss\_l2\_api.h.

### 5.272.2.3 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`

## 5.273 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`

### 5.273.1 Detailed Description

VLAN Control Entry.

Definition at line 1013 of file `vtss_l2_api.h`.

### 5.273.2 Field Documentation

#### 5.273.2.1 id

`vtss_vce_id_t` `vtss_vce_t::id`

VCE ID

Definition at line 1015 of file `vtss_l2_api.h`.

### 5.273.2.2 key

`vtss_vce_key_t` `vtss_vce_t::key`

VCE Key

Definition at line 1016 of file `vtss_l2_api.h`.

### 5.273.2.3 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`

## 5.274 `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`

### 5.274.1 Detailed Description

VCE tag information.

Definition at line 938 of file `vtss_l2_api.h`.

### 5.274.2 Field Documentation

#### 5.274.2.1 vid

`vtss_vcap_vid_t` `vtss_vce_tag_t::vid`

VLAN ID (12 bit)

Definition at line 940 of file `vtss_l2_api.h`.

#### 5.274.2.2 pcp

`vtss_vcap_u8_t` `vtss_vce_tag_t::pcp`

PCP (3 bit)

Definition at line 941 of file `vtss_l2_api.h`.

#### 5.274.2.3 dei

`vtss_vcap_bit_t` `vtss_vce_tag_t::dei`

DEI

Definition at line 942 of file `vtss_l2_api.h`.

#### 5.274.2.4 tagged

`vtss_vcap_bit_t` `vtss_vce_tag_t::tagged`

Tagged/untagged frame

Definition at line 943 of file `vtss_l2_api.h`.

#### 5.274.2.5 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`

## 5.275 vtss\_vcl\_port\_conf\_t Struct Reference

VCL port configuration.

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

### Data Fields

- `BOOL dmac_dip`

#### 5.275.1 Detailed Description

VCL port configuration.

Definition at line 878 of file vtss\_l2\_api.h.

#### 5.275.2 Field Documentation

##### 5.275.2.1 dmac\_dip

```
BOOL vtss_vcl_port_conf_t::dmac_dip
```

Enable DMAC/DIP matching (default SMAC/SIP)

Definition at line 879 of file vtss\_l2\_api.h.

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

- vtss\_api/include/[vtss\\_l2\\_api.h](#)

## 5.276 vtss\_vid\_mac\_t Struct Reference

MAC Address in specific VLAN.

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

### Data Fields

- `vtss_vid_t vid`
- `vtss_mac_t mac`

### 5.276.1 Detailed Description

MAC Address in specific VLAN.

Definition at line 652 of file types.h.

### 5.276.2 Field Documentation

#### 5.276.2.1 vid

`vtss_vid_t vtss_vid_mac_t::vid`

VLAN ID

Definition at line 654 of file types.h.

#### 5.276.2.2 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`

## 5.277 vtss\_vlan\_conf\_t Struct Reference

VLAN configuration.

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

### Data Fields

- `vtss_etype_t s_etype`

### 5.277.1 Detailed Description

VLAN configuration.

Definition at line 625 of file vtss\_l2\_api.h.

## 5.277.2 Field Documentation

### 5.277.2.1 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`

## 5.278 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`

### 5.278.1 Detailed Description

VLAN port configuration.

Definition at line 664 of file vtss\_l2\_api.h.

## 5.278.2 Field Documentation

### 5.278.2.1 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.

## 5.278.2.2 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.

## 5.278.2.3 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.

## 5.278.2.4 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.

## 5.278.2.5 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](#)

**5.279 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`

### 5.279.1 Detailed Description

VLAN tag with "arbitrary" TPID.

Definition at line 632 of file types.h.

### 5.279.2 Field Documentation

#### 5.279.2.1 tpid

`vtss_etype_t vtss_vlan_tag_t::tpid`

Tag Protocol Identifier

Definition at line 633 of file types.h.

#### 5.279.2.2 pcp

`vtss_tagprio_t vtss_vlan_tag_t::pcp`

Priority Code Point

Definition at line 634 of file types.h.

#### 5.279.2.3 dei

`BOOL vtss_vlan_tag_t::dei`

Drop Eligible Indicator

Definition at line 635 of file types.h.

## 5.279.2.4 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](#)

## 5.280 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](#)

### 5.280.1 Detailed Description

VLAN translation group-to-VLAN configuration.

Definition at line 1104 of file vtss\_l2\_api.h.

### 5.280.2 Field Documentation

#### 5.280.2.1 group\_id

```
u16 vtss_vlan_trans_grp2vlan_conf_t::group_id
```

Group ID

Definition at line 1105 of file vtss\_l2\_api.h.

### 5.280.2.2 vid

`vtss_vid_t vtss_vlan_trans_grp2vlan_conf_t::vid`

VLAN ID

Definition at line 1106 of file `vtss_l2_api.h`.

### 5.280.2.3 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`

## 5.281 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]`

### 5.281.1 Detailed Description

VLAN translation port-to-group configuration.

Definition at line 1098 of file `vtss_l2_api.h`.

### 5.281.2 Field Documentation

### 5.281.2.1 group\_id

```
u16 vtss_vlan_trans_port2grp_conf_t::group_id
```

Group ID

Definition at line 1099 of file vtss\_l2\_api.h.

### 5.281.2.2 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

## 5.282 vtss\_vlan\_vid\_conf\_t Struct Reference

VLAN ID configuration.

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

### Data Fields

- [BOOL learning](#)
- [BOOL mirror](#)

### 5.282.1 Detailed Description

VLAN ID configuration.

Definition at line 740 of file vtss\_l2\_api.h.

### 5.282.2 Field Documentation

### 5.282.2.1 learning

`BOOL vtss_vlan_vid_conf_t::learning`

Enable/disable learning

Definition at line 742 of file `vtss_l2_api.h`.

### 5.282.2.2 mirror

`BOOL vtss_vlan_vid_conf_t::mirror`

Enable/disable mirroring

Definition at line 743 of file `vtss_l2_api.h`.

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

- `vtss_api/include/vtss_l2_api.h`

## 5.283 vtss\_vstax\_conf\_t Struct Reference

VStaX configuration for switch.

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

### Data Fields

- `vtss_vstax_upsid_t upsid_0`
- `vtss_vstax_upsid_t upsid_1`
- `vtss_port_no_t port_0`
- `vtss_port_no_t port_1`
- `BOOL cmef_disable`

### 5.283.1 Detailed Description

VStaX configuration for switch.

Definition at line 2324 of file `vtss_l2_api.h`.

### 5.283.2 Field Documentation

### 5.283.2.1 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.

### 5.283.2.2 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.

### 5.283.2.3 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.

### 5.283.2.4 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.

### 5.283.2.5 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](#)

## 5.284 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`

#### 5.284.1 Detailed Description

GLAG info.

Definition at line 1611 of file vtss\_l2\_api.h.

#### 5.284.2 Field Documentation

##### 5.284.2.1 upsid

```
vtss_vstax_upsid_t vtss_vstax_glag_entry_t::upsid
```

UPS identifier

Definition at line 1613 of file vtss\_l2\_api.h.

##### 5.284.2.2 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](#)

## 5.285 vtss\_vstax\_port\_conf\_t Struct Reference

VStaX setup for port.

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

## Data Fields

- `u8 ttl`
- `BOOL mirror`

### 5.285.1 Detailed Description

VStaX setup for port.

Definition at line 2358 of file `vtss_l2_api.h`.

### 5.285.2 Field Documentation

#### 5.285.2.1 ttl

`u8 vtss_vstax_port_conf_t::ttl`

TTL, 0-31

Definition at line 2359 of file `vtss_l2_api.h`.

#### 5.285.2.2 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`

## 5.286 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`

### 5.286.1 Detailed Description

UPSID Route Entry.

Definition at line 2425 of file vtss\_l2\_api.h.

### 5.286.2 Field Documentation

#### 5.286.2.1 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.

#### 5.286.2.2 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](#)

## 5.287 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 \[VTSS\\_VSTAX\\_UPSIDS\]](#)

### 5.287.1 Detailed Description

UPSID Route Table.

Definition at line 2431 of file vtss\_l2\_api.h.

### 5.287.2 Field Documentation

#### 5.287.2.1 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.

#### 5.287.2.2 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

## 5.288 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](#)

### 5.288.1 Detailed Description

VStaX frame header used for reception.

Definition at line 190 of file vtss\_packet\_api.h.

### 5.288.2 Field Documentation

#### 5.288.2.1 valid

`BOOL vtss_vstax_rx_header_t::valid`

TRUE if frame was VStaX tagged

Definition at line 192 of file vtss\_packet\_api.h.

#### 5.288.2.2 sp

`BOOL vtss_vstax_rx_header_t::sp`

TRUE if received on super-prio

Definition at line 193 of file vtss\_packet\_api.h.

#### 5.288.2.3 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.

#### 5.288.2.4 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.

### 5.288.2.5 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.

### 5.288.2.6 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](#)

## 5.289 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

### 5.289.1 Detailed Description

VStaX frame header used for transmission.

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

## 5.289.2 Field Documentation

### 5.289.2.1 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.

### 5.289.2.2 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.

### 5.289.2.3 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.

### 5.289.2.4 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.

### 5.289.2.5 tci

`vtss_tci_t` `vtss_vstax_tx_header_t::tci`

VLAN tag information

Definition at line 459 of file vtss\_packet\_api.h.

### 5.289.2.6 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.

### 5.289.2.7 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.

### 5.289.2.8 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.

### 5.289.2.9 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.

### 5.289.2.10 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.

### 5.289.2.11 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`

## 5.290 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]`

### 5.290.1 Detailed Description

Structure for Wake-On-LAN MAC Address.

Definition at line 1656 of file `vtss_phy_api.h`.

### 5.290.2 Field Documentation

#### 5.290.2.1 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 6

## File Documentation

### 6.1 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.*

#### 6.1.1 Detailed Description

Layer 2 Public API Header for l2.

This header file describes public Layer 2 datatypes

#### 6.1.2 Enumeration Type Documentation

##### 6.1.2.1 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.

## 6.2 vtss\_api/include/vtss/api/options.h File Reference

Features and options.

### Macros

- #define VTSS\_ARCH\_JAGUAR\_2
- #define VTSS\_ARCH\_JAGUAR\_2\_B
- #define VTSS\_CHIP\_10G\_PHY
- #define VTSS\_FEATURE\_MISC
- #define VTSS\_FEATURE\_SERIAL\_GPIO
- #define VTSS\_FEATURE\_PORT\_CONTROL
- #define VTSS\_FEATURE\_PORT\_IHF
- #define VTSS\_FEATURE\_CLAUSE\_37
- #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\_VSTAX
- #define VTSS\_FEATURE\_VSTAX\_V2
- #define VTSS\_FEATURE\_AGGR\_GLAG
- #define VTSS\_FEATURE\_PORT\_MUX
- #define VTSS\_FEATURE\_PFC
- #define VTSS\_FEATURE\_QCL
- #define VTSS\_FEATURE\_QCL\_V2
- #define VTSS\_FEATURE\_QCL\_KEY\_S\_TAG
- #define VTSS\_FEATURE\_QCL\_KEY\_INNER\_TAG
- #define VTSS\_FEATURE\_QCL\_KEY\_DMAC
- #define VTSS\_FEATURE\_QCL\_KEY\_DIP
- #define VTSS\_FEATURE\_QCL\_PCP\_DEI\_ACTION
- #define VTSS\_FEATURE\_QCL\_POLICY\_ACTION
- #define VTSS\_FEATURE\_QOS\_POLICER\_UC\_SWITCH
- #define VTSS\_FEATURE\_QOS\_POLICER\_MC\_SWITCH
- #define VTSS\_FEATURE\_QOS\_POLICER\_BC\_SWITCH
- #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\_V2
- #define VTSS\_FEATURE\_QOS\_QUEUE\_POLICER
- #define VTSS\_FEATURE\_QOS\_QUEUE\_TX

- #define VTSS\_FEATURE\_QOS\_SCHEDULER\_V2
- #define VTSS\_FEATURE\_QOS\_SCHEDULER\_DWRR\_CNT
- #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\_SHAPERS\_DLBB
- #define VTSS\_FEATURE\_QOS\_DSCP\_CLASS\_DP\_AWARE
- #define VTSS\_FEATURE\_QOS\_DSCP\_REMARK
- #define VTSS\_FEATURE\_QOS\_DSCP\_REMARK\_V2
- #define VTSS\_FEATURE\_QOS\_WRED\_V3
- #define VTSS\_FEATURE\_QOS\_POLICER\_DLBB
- #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\_PVLAN
- #define VTSS\_FEATURE\_VLAN\_PORT\_V2
- #define VTSS\_FEATURE\_VLAN\_TX\_TAG
- #define VTSS\_FEATURE\_MAC\_AGE\_AUTO
- #define VTSS\_FEATURE\_MAC\_CPU\_QUEUE
- #define VTSS\_FEATURE\_EEE
- #define VTSS\_FEATURE\_FAN
- #define VTSS\_FEATURE\_VCAP
- #define VTSS\_FEATURE\_ACL
- #define VTSS\_FEATURE\_ACL\_V2
- #define VTSS\_FEATURE\_VCL
- #define VTSS\_FEATURE\_TIMESTAMP
- #define VTSS\_FEATURE\_PHY\_TIMESTAMP
- #define VTSS\_FEATURE\_TIMESTAMP\_ONE\_STEP
- #define VTSS\_FEATURE\_TIMESTAMP\_LATENCY\_COMP
- #define VTSS\_FEATURE\_TIMESTAMP\_ORG\_TIME
- #define VTSS\_FEATURE\_TIMESTAMP\_P2P\_DELAY\_COMP
- #define VTSS\_FEATURE\_TIMESTAMP\_ASYMMETRY\_COMP
- #define VTSS\_FEATURE\_SYNCE
- #define VTSS\_FEATURE\_NPI
- #define VTSS\_FEATURE\_LED\_POW\_REDUC
- #define VTSS\_FEATURE\_INTERRUPTS
- #define VTSS\_FEATURE\_IRQ\_CONTROL
- #define VTSS\_FEATURE\_VLAN\_TRANSLATION
- #define VTSS\_FEATURE\_SFLOW
- #define VTSS\_FEATURE\_SERDES\_MACRO\_SETTINGS
- #define VTSS\_FEATURE\_10GBASE\_KR
- #define VTSS\_FEATURE\_10G
- #define VTSS\_OPT\_VCORE\_III 1
- #define VTSS\_FEATURE\_FDMA
- #define VTSS\_OPT\_PCIE\_ACCESS
- #define VTSS\_PHY\_10G\_FIFO\_SYNC
- #define VIPER\_B\_FIFO\_RESET
- #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\_VERIPHYS 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

## 6.2.1 Detailed Description

Features and options.

This header file describes target features and compile-time options

## 6.2.2 Macro Definition Documentation

### 6.2.2.1 VTSS\_ARCH\_JAGUAR\_2

```
#define VTSS_ARCH_JAGUAR_2
```

Jaguar-2 architecture

Definition at line 207 of file options.h.

### 6.2.2.2 VTSS\_ARCH\_JAGUAR\_2\_B

```
#define VTSS_ARCH_JAGUAR_2_B
```

Jaguar-2 revision A/B architecture

Definition at line 208 of file options.h.

### 6.2.2.3 VTSS\_CHIP\_10G\_PHY

```
#define VTSS_CHIP_10G_PHY
```

10Gb 848x Phy API

Definition at line 213 of file options.h.

#### 6.2.2.4 VTSS\_FEATURE\_MISC

```
#define VTSS_FEATURE_MISC
```

Miscellaneous

Definition at line 214 of file options.h.

#### 6.2.2.5 VTSS\_FEATURE\_SERIAL\_GPIO

```
#define VTSS_FEATURE_SERIAL_GPIO
```

Serial GPIO control

Definition at line 215 of file options.h.

#### 6.2.2.6 VTSS\_FEATURE\_PORT\_CONTROL

```
#define VTSS_FEATURE_PORT_CONTROL
```

Port control

Definition at line 216 of file options.h.

#### 6.2.2.7 VTSS\_FEATURE\_PORT\_IFH

```
#define VTSS_FEATURE_PORT_IFH
```

Port IFH control

Definition at line 217 of file options.h.

#### 6.2.2.8 VTSS\_FEATURE\_CLAUSE\_37

```
#define VTSS_FEATURE_CLAUSE_37
```

IEEE 802.3 clause 37 auto-negotiation

Definition at line 218 of file options.h.

### 6.2.2.9 VTSS\_FEATURE\_EXC\_COL\_CONT

```
#define VTSS_FEATURE_EXC_COL_CONT
```

Excessive collision continuation

Definition at line 219 of file options.h.

### 6.2.2.10 VTSS\_FEATURE\_PORT\_CNT\_ETHER\_LIKE

```
#define VTSS_FEATURE_PORT_CNT_ETHER_LIKE
```

Ethernet-like counters

Definition at line 220 of file options.h.

### 6.2.2.11 VTSS\_FEATURE\_PORT\_CNT\_BRIDGE

```
#define VTSS_FEATURE_PORT_CNT_BRIDGE
```

Bridge counters

Definition at line 221 of file options.h.

### 6.2.2.12 VTSS\_FEATURE\_QOS

```
#define VTSS_FEATURE_QOS
```

QoS

Definition at line 222 of file options.h.

### 6.2.2.13 VTSS\_FEATURE\_VSTAX

```
#define VTSS_FEATURE_VSTAX
```

VStaX stacking

Definition at line 223 of file options.h.

### 6.2.2.14 VTSS\_FEATURE\_VSTAX\_V2

```
#define VTSS_FEATURE_VSTAX_V2
```

VStaX stacking, as implemented on Jaguar1 (VStaX2/AF)

Definition at line 224 of file options.h.

### 6.2.2.15 VTSS\_FEATURE\_AGGR\_GLAG

```
#define VTSS_FEATURE_AGGR_GLAG
```

Global link aggregations across stack

Definition at line 225 of file options.h.

### 6.2.2.16 VTSS\_FEATURE\_PORT\_MUX

```
#define VTSS_FEATURE_PORT_MUX
```

Port mux between serdes blocks and ports

Definition at line 226 of file options.h.

### 6.2.2.17 VTSS\_FEATURE\_PFC

```
#define VTSS_FEATURE_PFC
```

802.1Qbb Priority Flow Control

Definition at line 227 of file options.h.

### 6.2.2.18 VTSS\_FEATURE\_QCL

```
#define VTSS_FEATURE_QCL
```

QoS: QoS Control Lists

Definition at line 228 of file options.h.

### 6.2.2.19 VTSS FEATURE QCL V2

```
#define VTSS_FEATURE_QCL_V2
```

QoS: QoS Control Lists, V2 features

Definition at line 229 of file options.h.

### 6.2.2.20 VTSS FEATURE QCL KEY S TAG

```
#define VTSS_FEATURE_QCL_KEY_S_TAG
```

QoS: QoS Control Lists has S tag support

Definition at line 230 of file options.h.

### 6.2.2.21 VTSS FEATURE QCL KEY INNER TAG

```
#define VTSS_FEATURE_QCL_KEY_INNER_TAG
```

QoS: QoS Control Lists has inner tag

Definition at line 231 of file options.h.

### 6.2.2.22 VTSS FEATURE QCL KEY DMAC

```
#define VTSS_FEATURE_QCL_KEY_DMAC
```

QoS: QoS Control Lists has destination MAC address

Definition at line 232 of file options.h.

### 6.2.2.23 VTSS FEATURE QCL KEY DIP

```
#define VTSS_FEATURE_QCL_KEY_DIP
```

QoS: QoS Control Lists has destination IP address

Definition at line 233 of file options.h.

### 6.2.2.24 VTSS FEATURE QCL\_PCP\_DEI\_ACTION

```
#define VTSS_FEATURE_QCL_PCP_DEI_ACTION
```

QoS: QoS Control Lists has PCP and DEI action

Definition at line 234 of file options.h.

### 6.2.2.25 VTSS FEATURE QCL\_POLICY\_ACTION

```
#define VTSS_FEATURE_QCL_POLICY_ACTION
```

QoS: QoS Control Lists has policy action

Definition at line 235 of file options.h.

### 6.2.2.26 VTSS FEATURE QOS\_POLICER\_UC\_SWITCH

```
#define VTSS_FEATURE_QOS_POLICER_UC_SWITCH
```

QoS: Unicast policer per switch

Definition at line 236 of file options.h.

### 6.2.2.27 VTSS FEATURE QOS\_POLICER\_MC\_SWITCH

```
#define VTSS_FEATURE_QOS_POLICER_MC_SWITCH
```

QoS: Multicast policer per switch

Definition at line 237 of file options.h.

### 6.2.2.28 VTSS FEATURE QOS\_POLICER\_BC\_SWITCH

```
#define VTSS_FEATURE_QOS_POLICER_BC_SWITCH
```

QoS: Broadcast policer per switch

Definition at line 238 of file options.h.

### 6.2.2.29 VTSS FEATURE QOS PORT POLICER EXT

```
#define VTSS_FEATURE_QOS_PORT_POLICER_EXT
```

QoS: Port Policer Extensions

Definition at line 239 of file options.h.

### 6.2.2.30 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 240 of file options.h.

### 6.2.2.31 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 241 of file options.h.

### 6.2.2.32 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 242 of file options.h.

### 6.2.2.33 VTSS FEATURE QOS PORT POLICER EXT TTM V2

```
#define VTSS_FEATURE_QOS_PORT_POLICER_EXT_TTM_V2
```

QoS: Port Policer has Traffic Type Mask version 2 support

Definition at line 243 of file options.h.

### 6.2.2.34 VTSS\_FEATURE\_QOS\_QUEUE\_POLICER

```
#define VTSS_FEATURE_QOS_QUEUE_POLICER
```

QoS: Has Ingress Queue Policers

Definition at line 244 of file options.h.

### 6.2.2.35 VTSS\_FEATURE\_QOS\_QUEUE\_TX

```
#define VTSS_FEATURE_QOS_QUEUE_TX
```

QoS: Has TX Queue support

Definition at line 245 of file options.h.

### 6.2.2.36 VTSS\_FEATURE\_QOS\_SCHEDULER\_V2

```
#define VTSS_FEATURE_QOS_SCHEDULER_V2
```

QoS: 2. version of scheduler

Definition at line 246 of file options.h.

### 6.2.2.37 VTSS\_FEATURE\_QOS\_SCHEDULER\_DWRR\_CNT

```
#define VTSS_FEATURE_QOS_SCHEDULER_DWRR_CNT
```

QoS: Scheduler supports variable number of DWRR inputs

Definition at line 247 of file options.h.

### 6.2.2.38 VTSS\_FEATURE\_QOS\_TAG\_REMARK\_V2

```
#define VTSS_FEATURE_QOS_TAG_REMARK_V2
```

QoS: 2. version of tag priority remarking

Definition at line 248 of file options.h.

### 6.2.2.39 VTSS\_FEATURE\_QOS\_CLASSIFICATION\_V2

```
#define VTSS_FEATURE_QOS_CLASSIFICATION_V2
```

QoS: 2. version of classification

Definition at line 249 of file options.h.

### 6.2.2.40 VTSS\_FEATURE\_QOS\_EGRESS\_QUEUE\_SHAPERS

```
#define VTSS_FEATURE_QOS_EGRESS_QUEUE_SHAPERS
```

QoS: Has Egress Queue Shapers

Definition at line 250 of file options.h.

### 6.2.2.41 VTSS\_FEATURE\_QOS\_EGRESS\_SHAPERS\_DLBB

```
#define VTSS_FEATURE_QOS_EGRESS_SHAPERS_DLBB
```

QoS: Egress shapers has DLB support

Definition at line 251 of file options.h.

### 6.2.2.42 VTSS\_FEATURE\_QOS\_DSCP\_CLASS\_DP\_AWARE

```
#define VTSS_FEATURE_QOS_DSCP_CLASS_DP_AWARE
```

QoS: DSCP classification is DP aware

Definition at line 252 of file options.h.

### 6.2.2.43 VTSS\_FEATURE\_QOS\_DSCP\_REMARK

```
#define VTSS_FEATURE_QOS_DSCP_REMARK
```

QoS: Has DSCP remarking

Definition at line 253 of file options.h.

#### 6.2.2.44 VTSS\_FEATURE\_QOS\_DSCP\_REMARK\_V2

```
#define VTSS_FEATURE_QOS_DSCP_REMARK_V2
```

QoS: 2. version of DSCP remarking

Definition at line 254 of file options.h.

#### 6.2.2.45 VTSS\_FEATURE\_QOS\_WRED\_V3

```
#define VTSS_FEATURE_QOS_WRED_V3
```

QoS: WRED global - per queue (0..7), per dpl (1..3), per group (0..2)

Definition at line 255 of file options.h.

#### 6.2.2.46 VTSS\_FEATURE\_QOS\_POLICER\_DLDB

```
#define VTSS_FEATURE_QOS_POLICER_DLDB
```

DLB policers

Definition at line 256 of file options.h.

#### 6.2.2.47 VTSS\_FEATURE\_PACKET

```
#define VTSS_FEATURE_PACKET
```

CPU Rx/Tx frame configuration

Definition at line 257 of file options.h.

#### 6.2.2.48 VTSS\_FEATURE\_PACKET\_TX

```
#define VTSS_FEATURE_PACKET_TX
```

CPU Tx frame

Definition at line 258 of file options.h.

### 6.2.2.49 VTSS FEATURE PACKET\_RX

```
#define VTSS_FEATURE_PACKET_RX
```

CPU Rx frame

Definition at line 259 of file options.h.

### 6.2.2.50 VTSS FEATURE PACKET\_GROUPING

```
#define VTSS_FEATURE_PACKET_GROUPING
```

Extraction and injection occurs through extraction and injection groups rather than queues.

Definition at line 260 of file options.h.

### 6.2.2.51 VTSS FEATURE PACKET\_PORT\_REG

```
#define VTSS_FEATURE_PACKET_PORT_REG
```

Packet registration per port

Definition at line 261 of file options.h.

### 6.2.2.52 VTSS FEATURE LAYER2

```
#define VTSS_FEATURE_LAYER2
```

Layer 2 (switching)

Definition at line 262 of file options.h.

### 6.2.2.53 VTSS FEATURE\_PVLAN

```
#define VTSS_FEATURE_PVLAN
```

Private VLANs

Definition at line 263 of file options.h.

**6.2.2.54 VTSS\_FEATURE\_VLAN\_PORT\_V2**

```
#define VTSS_FEATURE_VLAN_PORT_V2
```

VLAN port configuration, V2 features

Definition at line 264 of file options.h.

**6.2.2.55 VTSS\_FEATURE\_VLAN\_TX\_TAG**

```
#define VTSS_FEATURE_VLAN_TX_TAG
```

VLAN tagging per (VID, port)

Definition at line 265 of file options.h.

**6.2.2.56 VTSS\_FEATURE\_MAC\_AGE\_AUTO**

```
#define VTSS_FEATURE_MAC_AGE_AUTO
```

Automatic MAC address ageing

Definition at line 266 of file options.h.

**6.2.2.57 VTSS\_FEATURE\_MAC\_CPU\_QUEUE**

```
#define VTSS_FEATURE_MAC_CPU_QUEUE
```

CPU queue per MAC address

Definition at line 267 of file options.h.

**6.2.2.58 VTSS\_FEATURE\_EEE**

```
#define VTSS_FEATURE_EEE
```

Energy Efficient Ethernet

Definition at line 269 of file options.h.

### 6.2.2.59 VTSS\_FEATURE\_FAN

```
#define VTSS_FEATURE_FAN
```

Fan control

Definition at line 270 of file options.h.

### 6.2.2.60 VTSS\_FEATURE\_VCAP

```
#define VTSS_FEATURE_VCAP
```

VCAP

Definition at line 271 of file options.h.

### 6.2.2.61 VTSS\_FEATURE\_ACL

```
#define VTSS_FEATURE_ACL
```

Access Control Lists

Definition at line 272 of file options.h.

### 6.2.2.62 VTSS\_FEATURE\_ACL\_V2

```
#define VTSS_FEATURE_ACL_V2
```

Access Control Lists, V2 features

Definition at line 273 of file options.h.

### 6.2.2.63 VTSS\_FEATURE\_VCL

```
#define VTSS_FEATURE_VCL
```

VLAN Control Lists

Definition at line 274 of file options.h.

### 6.2.2.64 VTSS\_FEATURE\_TIMESTAMP

```
#define VTSS_FEATURE_TIMESTAMP
```

Packet timestamp feature (for PTP and OAM)

Definition at line 275 of file options.h.

### 6.2.2.65 VTSS\_FEATURE\_PHY\_TIMESTAMP

```
#define VTSS_FEATURE_PHY_TIMESTAMP
```

PHY timestamp feature (for PTP/OAM)

Definition at line 276 of file options.h.

### 6.2.2.66 VTSS\_FEATURE\_TIMESTAMP\_ONE\_STEP

```
#define VTSS_FEATURE_TIMESTAMP_ONE_STEP
```

ONESTEP timestamp hardware support

Definition at line 277 of file options.h.

### 6.2.2.67 VTSS\_FEATURE\_TIMESTAMP\_LATENCY\_COMP

```
#define VTSS_FEATURE_TIMESTAMP_LATENCY_COMP
```

Ingress and egress latency compensation hardware support

Definition at line 278 of file options.h.

### 6.2.2.68 VTSS\_FEATURE\_TIMESTAMP\_ORG\_TIME

```
#define VTSS_FEATURE_TIMESTAMP_ORG_TIME
```

OriginTimestamp update hardware support

Definition at line 279 of file options.h.

### 6.2.2.69 VTSS FEATURE\_TIMESTAMP\_P2P\_DELAY\_COMP

```
#define VTSS_FEATURE_TIMESTAMP_P2P_DELAY_COMP
```

Peer-to-peer path delay compensation hardware support

Definition at line 280 of file options.h.

### 6.2.2.70 VTSS FEATURE\_TIMESTAMP\_ASYMMETRY\_COMP

```
#define VTSS_FEATURE_TIMESTAMP_ASYMMETRY_COMP
```

Path delay asymmetry compensation hardware support

Definition at line 281 of file options.h.

### 6.2.2.71 VTSS FEATURE\_SYNCE

```
#define VTSS_FEATURE_SYNCE
```

SYNCE - L1 synchronization feature

Definition at line 283 of file options.h.

### 6.2.2.72 VTSS FEATURE\_NPI

```
#define VTSS_FEATURE_NPI
```

NPI port

Definition at line 284 of file options.h.

### 6.2.2.73 VTSS FEATURE\_LED\_POW\_REDUC

```
#define VTSS_FEATURE_LED_POW_REDUC
```

LED power reduction

Definition at line 285 of file options.h.

### 6.2.2.74 VTSS\_FEATURE\_INTERRUPTS

```
#define VTSS_FEATURE_INTERRUPTS
```

Port Interrupt support

Definition at line 286 of file options.h.

### 6.2.2.75 VTSS\_FEATURE\_IRQ\_CONTROL

```
#define VTSS_FEATURE_IRQ_CONTROL
```

General IRQ support

Definition at line 287 of file options.h.

### 6.2.2.76 VTSS\_FEATURE\_VLAN\_TRANSLATION

```
#define VTSS_FEATURE_VLAN_TRANSLATION
```

VLAN Translation

Definition at line 288 of file options.h.

### 6.2.2.77 VTSS\_FEATURE\_SFLOW

```
#define VTSS_FEATURE_SFLOW
```

sFlow feature

Definition at line 289 of file options.h.

### 6.2.2.78 VTSS\_FEATURE\_SERDES\_MACRO\_SETTINGS

```
#define VTSS_FEATURE_SERDES_MACRO_SETTINGS
```

Hooks for Serdes Macro configuration

Definition at line 291 of file options.h.

### 6.2.2.79 VTSS\_FEATURE\_10GBASE\_KR

```
#define VTSS_FEATURE_10GBASE_KR
```

KR

Definition at line 292 of file options.h.

### 6.2.2.80 VTSS\_FEATURE\_10G

```
#define VTSS_FEATURE_10G
```

10G ports

Definition at line 295 of file options.h.

### 6.2.2.81 VTSS\_OPT\_VCORE\_III

```
#define VTSS_OPT_VCORE_III 1
```

Internal VCORE-III (MIPS) CPU enabled by default

Definition at line 297 of file options.h.

### 6.2.2.82 VTSS\_FEATURE\_FDMA

```
#define VTSS_FEATURE_FDMA
```

Frame DMA

Definition at line 300 of file options.h.

### 6.2.2.83 VTSS\_OPT\_PCIE\_ACCESS

```
#define VTSS_OPT_PCIE_ACCESS
```

PCIe access from external CPU

Definition at line 309 of file options.h.

**6.2.2.84 VTSS\_PHY\_10G\_FIFO\_SYNC**

```
#define VTSS_PHY_10G_FIFO_SYNC
```

TS FIFO SYNC For 10G PHY

Definition at line 311 of file options.h.

**6.2.2.85 VIPER\_B\_FIFO\_RESET**

```
#define VIPER_B_FIFO_RESET
```

Viper B 1588 FIFO sync

Definition at line 312 of file options.h.

**6.2.2.86 VTSS\_CHIP CU PHY**

```
#define VTSS_CHIP CU PHY
```

Copper PHY chip

Definition at line 540 of file options.h.

**6.2.2.87 VTSS\_OPT\_TRACE**

```
#define VTSS_OPT_TRACE 1
```

Trace enabled by default

Definition at line 548 of file options.h.

**6.2.2.88 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.

### 6.2.2.89 VTSS\_OPT\_FDMA\_DEBUG

```
#define VTSS_OPT_FDMA_DEBUG 0
```

FDMA debug disabled by default

Definition at line 599 of file options.h.

### 6.2.2.90 VTSS\_OPT\_VAUI\_EQ\_CTRL

```
#define VTSS_OPT_VAUI_EQ_CTRL 6
```

Default equalization control

Definition at line 605 of file options.h.

### 6.2.2.91 VTSS\_PHY\_OPT\_VERIPHY

```
#define VTSS_PHY_OPT_VERIPHY 1
```

VeriPHY enabled by default

Definition at line 613 of file options.h.

### 6.2.2.92 VTSS\_FEATURE\_WARM\_START [1/2]

```
#define VTSS_FEATURE_WARM_START
```

Warm start

Definition at line 624 of file options.h.

### 6.2.2.93 VTSS\_FEATURE\_SYNCE\_10G

```
#define VTSS_FEATURE_SYNCE_10G
```

SYNCE - L1 synchronization feature for 10G PHYs

Definition at line 621 of file options.h.

### 6.2.2.94 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.

### 6.2.2.95 VTSS\_FEATURE\_WIS

```
#define VTSS_FEATURE_WIS
```

WAN interface sublayer functionality

Definition at line 623 of file options.h.

### 6.2.2.96 VTSS\_FEATURE\_WARM\_START [2/2]

```
#define VTSS_FEATURE_WARM_START
```

Warm start

Definition at line 624 of file options.h.

### 6.2.2.97 VTSS\_ARCH\_MALIBU

```
#define VTSS_ARCH_MALIBU
```

Used for Malibu-A PHY

Definition at line 625 of file options.h.

### 6.2.2.98 VTSS\_ARCH\_MALIBU\_B

```
#define VTSS_ARCH_MALIBU_B
```

Used for Malibu-B PHY

Definition at line 626 of file options.h.

### 6.2.2.99 VTSS\_ARCH\_VENICE\_C

```
#define VTSS_ARCH_VENICE_C
```

Used for Venice-C PHY

Definition at line 627 of file options.h.

## 6.3 vtss\_api/include/vtss/api/phy.h File Reference

PHY Public API Header.

```
#include <vtss/api/types.h>
```

### Macros

- `#define VTSS_PHY_POWER_ACTIPHYSY_BIT 0`
- `#define VTSS_PHY_POWER_DYNAMIC_BIT 1`

### Enumerations

- enum `vtss_phy_power_mode_t` { `VTSS_PHY_POWER_NOMINAL` = 0, `VTSS_PHY_POWER_ACTIPHYSY` = 1 << `VTSS_PHY_POWER_ACTIPHYSY_BIT`, `VTSS_PHY_POWER_DYNAMIC` = 1 << `VTSS_PHY_POWER_DYNAMIC_BIT`, `VTSS_PHY_POWER_ENABLED` = `VTSS_PHY_POWER_ACTIPHYSY` + `VTSS_PHY_POWER_DYNAMIC` }
- PHY power reduction modes.*
- enum `vtss_phy_veriphy_status_t` {  
`VTSS_VERIPHYSY_STATUS_OK` = 0, `VTSS_VERIPHYSY_STATUS_OPEN` = 1, `VTSS_VERIPHYSY_STATUS_SHORT` = 2, `VTSS_VERIPHYSY_STATUS_ABNORM` = 4,  
`VTSS_VERIPHYSY_STATUS_SHORT_A` = 8, `VTSS_VERIPHYSY_STATUS_SHORT_B` = 9, `VTSS_VERIPHYSY_STATUS_SHORT_C` = 10, `VTSS_VERIPHYSY_STATUS_SHORT_D` = 11,  
`VTSS_VERIPHYSY_STATUS_COUPL_A` = 12, `VTSS_VERIPHYSY_STATUS_COUPL_B` = 13, `VTSS_VERIPHYSY_STATUS_COUPL_C` = 14, `VTSS_VERIPHYSY_STATUS_COUPL_D` = 15,  
`VTSS_VERIPHYSY_STATUS_UNKNOWN` = 16, `VTSS_VERIPHYSY_STATUS_RUNNING` = 17 }
- VeriPHY status.*

### 6.3.1 Detailed Description

PHY Public API Header.

This header file describes public PHY data-types

### 6.3.2 Macro Definition Documentation

### 6.3.2.1 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.

### 6.3.2.2 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.

## 6.3.3 Enumeration Type Documentation

### 6.3.3.1 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.

### 6.3.3.2 vtss\_phy\_veriphy\_status\_t

```
enum vtss_phy_veriphy_status_t
```

VeriPHY status.

Enumerator

VTSS_VERIPHYS_STATUS_OK	Correctly terminated pair
-------------------------	---------------------------

## Enumerator

VTSS_VERIPHY_STATUS_OPEN	Open pair
VTSS_VERIPHY_STATUS_SHORT	Short pair
VTSS_VERIPHY_STATUS_ABNORM	Abnormal termination
VTSS_VERIPHY_STATUS_SHORT_A	Cross-pair short to pair A
VTSS_VERIPHY_STATUS_SHORT_B	Cross-pair short to pair B
VTSS_VERIPHY_STATUS_SHORT_C	Cross-pair short to pair C
VTSS_VERIPHY_STATUS_SHORT_D	Cross-pair short to pair D
VTSS_VERIPHY_STATUS_COUPL_A	Abnormal cross-pair coupling, pair A
VTSS_VERIPHY_STATUS_COUPL_B	Abnormal cross-pair coupling, pair B
VTSS_VERIPHY_STATUS_COUPL_C	Abnormal cross-pair coupling, pair C
VTSS_VERIPHY_STATUS_COUPL_D	Abnormal cross-pair coupling, pair D
VTSS_VERIPHY_STATUS_UNKNOWN	Unknown - VeriPhy never started ?
VTSS_VERIPHY_STATUS_RUNNING	VeriPhy is still running - No result yet

Definition at line 54 of file phy.h.

## 6.4 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 | PORT\_CAP\_DUAL\_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)
- #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.*

### 6.4.1 Detailed Description

Port Public API Header.

This header file describes public port data-types

### 6.4.2 Macro Definition Documentation

#### 6.4.2.1 PORT\_CAP\_NONE

```
#define PORT_CAP_NONE 0x00000000
```

No capabilities

Definition at line 46 of file port.h.

#### 6.4.2.2 PORT\_CAP\_AUTONEG

```
#define PORT_CAP_AUTONEG 0x00000001
```

Auto negotiation

Definition at line 47 of file port.h.

#### 6.4.2.3 PORT\_CAP\_10M\_HDX

```
#define PORT_CAP_10M_HDX 0x00000002
```

10 Mbps, half duplex

Definition at line 48 of file port.h.

#### 6.4.2.4 PORT\_CAP\_10M\_FDX

```
#define PORT_CAP_10M_FDX 0x00000004
```

10 Mbps, full duplex

Definition at line 49 of file port.h.

#### 6.4.2.5 PORT\_CAP\_100M\_HDX

```
#define PORT_CAP_100M_HDX 0x00000008
```

100 Mbps, half duplex

Definition at line 50 of file port.h.

#### 6.4.2.6 PORT\_CAP\_100M\_FDX

```
#define PORT_CAP_100M_FDX 0x00000010
```

100 Mbps, full duplex

Definition at line 51 of file port.h.

#### 6.4.2.7 PORT\_CAP\_1G\_FDX

```
#define PORT_CAP_1G_FDX 0x00000020
```

1 Gbps, full duplex

Definition at line 52 of file port.h.

#### 6.4.2.8 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.

#### 6.4.2.9 PORT\_CAP\_5G\_FDX

```
#define PORT_CAP_5G_FDX 0x00000080
```

5Gbps, full duplex

Definition at line 54 of file port.h.

#### 6.4.2.10 PORT\_CAP\_10G\_FDX

```
#define PORT_CAP_10G_FDX 0x00000100
```

10Gbps, full duplex

Definition at line 55 of file port.h.

#### 6.4.2.11 PORT\_CAP\_FLOW\_CTRL

```
#define PORT_CAP_FLOW_CTRL 0x00001000
```

Flow control

Definition at line 56 of file port.h.

#### 6.4.2.12 PORT\_CAP\_COPPER

```
#define PORT_CAP_COPPER 0x00002000
```

Copper media

Definition at line 57 of file port.h.

#### 6.4.2.13 PORT\_CAP\_FIBER

```
#define PORT_CAP_FIBER 0x00004000
```

Fiber media

Definition at line 58 of file port.h.

#### 6.4.2.14 PORT\_CAP\_DUAL\_COPPER

```
#define PORT_CAP_DUAL_COPPER 0x00008000
```

Dual media, copper preferred

Definition at line 59 of file port.h.

#### 6.4.2.15 PORT\_CAP\_DUAL\_FIBER

```
#define PORT_CAP_DUAL_FIBER 0x00010000
```

Dual media, fiber preferred

Definition at line 60 of file port.h.

#### 6.4.2.16 PORT\_CAP\_SD\_ENABLE

```
#define PORT_CAP_SD_ENABLE 0x00020000
```

Signal Detect enabled

Definition at line 61 of file port.h.

#### 6.4.2.17 PORT\_CAP\_SD\_HIGH

```
#define PORT_CAP_SD_HIGH 0x00040000
```

Signal Detect active high

Definition at line 62 of file port.h.

#### 6.4.2.18 PORT\_CAP\_SD\_INTERNAL

```
#define PORT_CAP_SD_INTERNAL 0x00080000
```

Signal Detect select internal

Definition at line 63 of file port.h.

#### 6.4.2.19 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.

#### 6.4.2.20 PORT\_CAP\_XAUI\_LANE\_FLIP

```
#define PORT_CAP_XAUI_LANE_FLIP 0x00200000
```

Flip the XAUI lanes

Definition at line 65 of file port.h.

#### 6.4.2.21 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.

#### 6.4.2.22 PORT\_CAP\_SFP\_DETECT

```
#define PORT_CAP_SFP_DETECT 0x00800000
```

Auto detect the SFP module

Definition at line 67 of file port.h.

#### 6.4.2.23 PORT\_CAP\_STACKING

```
#define PORT_CAP_STACKING 0x01000000
```

Stack port candidate

Definition at line 68 of file port.h.

#### 6.4.2.24 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.

#### 6.4.2.25 PORT\_CAP\_SFP\_ONLY

```
#define PORT_CAP_SFP_ONLY 0x04000000
```

SFP only port (not dual media)

Definition at line 70 of file port.h.

#### 6.4.2.26 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.

#### 6.4.2.27 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.

#### 6.4.2.28 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.

#### 6.4.2.29 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.

#### 6.4.2.30 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.

#### 6.4.2.31 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.

#### 6.4.2.32 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.

#### 6.4.2.33 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.

#### 6.4.2.34 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.

#### 6.4.2.35 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.

#### 6.4.2.36 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.

#### 6.4.2.37 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.

#### 6.4.2.38 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.

#### 6.4.2.39 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.

#### 6.4.2.40 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_100FX  
| PORT_CAP_DUAL_SFP_DETECT)
```

Copper & Fiber mode, but SFP auto detection not supported

Definition at line 88 of file port.h.

#### 6.4.2.41 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.

#### 6.4.2.42 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.

#### 6.4.2.43 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.

#### 6.4.2.44 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.

#### 6.4.2.45 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.

#### 6.4.2.46 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.

#### 6.4.2.47 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.

---

### 6.4.3 Typedef Documentation

#### 6.4.3.1 port\_cap\_t

```
typedef u32 port_cap_t
```

Bit-mask containing the port capabilities

Definition at line 99 of file port.h.

### 6.4.4 Enumeration Type Documentation

#### 6.4.4.1 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.

#### 6.4.4.2 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
--	--

## Enumerator

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.

## 6.5 vtss\_api/include/vtss/api/types.h File Reference

Generic types API.

```
#include <vtss/api/options.h>
#include <sys/bsdtypes.h>
```

### Data Structures

- struct [vtss\\_aneg\\_t](#)  
*Auto negotiation struct.*
- struct [vtss\\_vlan\\_tag\\_t](#)
- struct [vtss\\_mac\\_t](#)  
*MAC Address.*
- struct [vtss\\_vid\\_mac\\_t](#)  
*MAC Address in specific VLAN.*
- struct [vtss\\_packet\\_rx\\_port\\_conf\\_t](#)  
*Packet registration per port.*
- struct [vtss\\_ipv6\\_t](#)  
*IPv6 address/mask.*
- struct [vtss\\_ip\\_addr\\_t](#)  
*Either an IPv4 or IPv6 address.*
- struct [vtss\\_ipv4\\_network\\_t](#)  
*IPv4 network.*
- struct [vtss\\_ipv6\\_network\\_t](#)  
*IPv6 network.*
- struct [vtss\\_ip\\_network\\_t](#)  
*IPv6 network.*
- struct [vtss\\_ipv4\\_uc\\_t](#)  
*IPv4 unicast routing entry.*
- struct [vtss\\_ipv6\\_uc\\_t](#)  
*IPv6 routing entry.*
- struct [vtss\\_routing\\_entry\\_t](#)  
*Routing entry.*
- struct [vtss\\_l3\\_counters\\_t](#)  
*Routing interface statics counter.*
- struct [vtss\\_vcap\\_u8\\_t](#)

- struct [vtss\\_vcap\\_u16\\_t](#)  
VCAP 8 bit value and mask.
- struct [vtss\\_vcap\\_u24\\_t](#)  
VCAP 16 bit value and mask.
- struct [vtss\\_vcap\\_u32\\_t](#)  
VCAP 24 bit value and mask.
- struct [vtss\\_vcap\\_u40\\_t](#)  
VCAP 32 bit value and mask.
- struct [vtss\\_vcap\\_u48\\_t](#)  
VCAP 40 bit value and mask.
- struct [vtss\\_vcap\\_u128\\_t](#)  
VCAP 48 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\\_timestamp\\_t](#)  
*Time stamp in seconds and nanoseconds.*

## Macros

- #define [PRIu64](#) "llu"
- #define [PRIi64](#) "lli"
- #define [PRIx64](#) "llx"
- #define [VTSS\\_BIT64](#)(x) (1ULL << (x))
- #define [VTSS\\_BITMASK64](#)(x) ((1ULL << (x)) - 1)
- #define [VTSS\\_EXTRACT\\_BITFIELD64](#)(x, o, w) (((x) >> (o)) & [VTSS\\_BITMASK64](#)(w))
- #define [VTSS\\_ENCODE\\_BITFIELD64](#)(x, o, w) (((u64)(x) & [VTSS\\_BITMASK64](#)(w)) << (o))
- #define [VTSS\\_ENCODE\\_BITMASK64](#)(o, w) ([VTSS\\_BITMASK64](#)(w) << (o))
- #define [TRUE](#) 1
- #define [FALSE](#) 0
- #define [VTSS\\_PACKET\\_RATE\\_DISABLED](#) 0xffffffff
- #define [VTSS\\_PORT\\_COUNT](#) 1
- #define [VTSS\\_PORT\\_COUNT](#) 53
- #define [VTSS\\_PORTS](#) [VTSS\\_OPT\\_PORT\\_COUNT](#)
- #define [VTSS\\_PORT\\_NO\\_NONE](#) (0xffffffff)
- #define [VTSS\\_PORT\\_NO\\_CPU](#) (0xfffffffffe)
- #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 256
- #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 2
- #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 8
- #define VTSS\_PACKET\_RX\_GRP\_CNT 2
- #define VTSS\_PACKET\_TX\_GRP\_CNT 2
- #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 0x7fffffffffffffffLL
- #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\_SYNCE\_CLK\_PORT\_ARRAY\_SIZE 4

## 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 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\_MACADDR** = -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\_USE** = -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\_USE** = -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\_VALID** = -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_INTERFACE`,
`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` }

*Serdess macro mode.*

- enum `vtss_storm_policer_mode_t` { `VTSS_STORM_POLICER_MODE_PORTS_AND_CPU`, `VTSS_STORM_POLICER_MODE_PORTS`, `VTSS_STORM_POLICER_MODE_CPU_ONLY` }

*Storm policer mode configuration.*

- enum `vtss_policer_type_t` { `VTSS_POLICER_TYPE_MEF`, `VTSS_POLICER_TYPE_SINGLE` }

*Dual leaky buckets policer configuration.*

- enum `vtss_vlan_frame_t` { `VTSS_VLAN_FRAME_ALL`, `VTSS_VLAN_FRAME_TAGGED`, `VTSS_VLAN_FRAME_UNTAGGED` }

*VLAN acceptable frame type.*

- enum `vtss_packet_reg_type_t` {
`VTSS_PACKET_REG_NORMAL`, `VTSS_PACKET_REG_FORWARD`, `VTSS_PACKET_REG_DISCARD`,
`VTSS_PACKET_REG_CPU_COPY`,
`VTSS_PACKET_REG_CPU_ONLY` }

*Packet registration type.*

- enum `vtss_vdd_t` { `VTSS_VDD_1V0`, `VTSS_VDD_1V2` }

*VDD power supply.*

- enum `vtss_ip_type_t` { `VTSS_IP_TYPE_NONE` = 0, `VTSS_IP_TYPE_IPV4` = 1, `VTSS_IP_TYPE_IPV6` = 2 }

*IP address type.*

- enum `vtss_routing_entry_type_t` { `VTSS_ROUTING_ENTRY_TYPE_INVALID` = 0, `VTSS_ROUTING_ENTRY_TYPE_IPV6_UC` = 1, `VTSS_ROUTING_ENTRY_TYPE_IPV4_MC` = 2, `VTSS_ROUTING_ENTRY_TYPE_IPV4_UC` = 3 }

*Routing entry type.*

- enum `vtss_vcap_bit_t` { `VTSS_VCAP_BIT_ANY`, `VTSS_VCAP_BIT_0`, `VTSS_VCAP_BIT_1` }

*VCAP 1 bit.*

- enum `vtss_vcap_vr_type_t` { `VTSS_VCAP_VR_TYPE_VALUE_MASK`, `VTSS_VCAP_VR_TYPE_RANGE_INCLUSIVE`, `VTSS_VCAP_VR_TYPE_RANGE_EXCLUSIVE` }

*Value/Range type.*

- enum `vtss_vcap_key_type_t` { `VTSS_VCAP_KEY_TYPE_NORMAL`, `VTSS_VCAP_KEY_TYPE_DOUBLE_TAG`, `VTSS_VCAP_KEY_TYPE_IP_ADDR`, `VTSS_VCAP_KEY_TYPE_MAC_IP_ADDR` }

- VCAP key type.*
- enum `vtss_hqos_sch_mode_t`{`VTSS_HQOS_SCH_MODE_NORMAL`, `VTSS_HQOS_SCH_MODE_BASIC`, `VTSS_HQOS_SCH_MODE_HIERARCHICAL`}
- HQoS port scheduling mode.*

### 6.5.1 Detailed Description

Generic types API.

This header file describes generic types used in the API

### 6.5.2 Macro Definition Documentation

#### 6.5.2.1 PRIu64

```
#define PRIu64 "llu"
```

Fallback un-signed 64-bit formatting string

Definition at line 111 of file types.h.

#### 6.5.2.2 PRIi64

```
#define PRIi64 "lli"
```

Fallback signed 64-bit formatting string

Definition at line 115 of file types.h.

#### 6.5.2.3 PRIx64

```
#define PRIx64 "llx"
```

Fallback hex 64-bit formatting string

Definition at line 119 of file types.h.

### 6.5.2.4 VTSS\_BIT64

```
#define VTSS_BIT64(  
    x ) (1ULL << (x))
```

Set one bit in a 64-bit mask

Definition at line 122 of file types.h.

### 6.5.2.5 VTSS\_BITMASK64

```
#define VTSS_BITMASK64(  
    x ) ((1ULL << (x)) - 1)
```

Get a bitmask consisting of x ones

Definition at line 123 of file types.h.

### 6.5.2.6 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.

### 6.5.2.7 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.

### 6.5.2.8 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.

### 6.5.2.9 TRUE

```
#define TRUE 1
```

True boolean value

Definition at line 129 of file types.h.

### 6.5.2.10 FALSE

```
#define FALSE 0
```

False boolean value

Definition at line 132 of file types.h.

### 6.5.2.11 VTSS\_PACKET\_RATE\_DISABLED

```
#define VTSS_PACKET_RATE_DISABLED 0xffffffff
```

Special value for disabling packet policer

Definition at line 238 of file types.h.

### 6.5.2.12 VTSS\_PORT\_COUNT [1/2]

```
#define VTSS_PORT_COUNT 1
```

Default number of ports

Number of ports

Definition at line 436 of file types.h.

### 6.5.2.13 VTSS\_PORT\_COUNT [2/2]

```
#define VTSS_PORT_COUNT 53
```

Default number of ports

Number of ports

Definition at line 436 of file types.h.

### 6.5.2.14 VTSS\_PORTS

```
#define VTSS_PORTS VTSS_OPT_PORT_COUNT
```

Number of ports

Definition at line 442 of file types.h.

### 6.5.2.15 VTSS\_PORT\_NO\_NONE

```
#define VTSS_PORT_NO_NONE (0xffffffff)
```

Port number none

Definition at line 448 of file types.h.

### 6.5.2.16 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.

### 6.5.2.17 VTSS\_PORT\_NO\_START

```
#define VTSS_PORT_NO_START (0)
```

Port start number

Definition at line 450 of file types.h.

### 6.5.2.18 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.

### 6.5.2.19 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.

### 6.5.2.20 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.

### 6.5.2.21 VTSS\_PRIOS

```
#define VTSS_PRIOS 8
```

Number of priorities

Definition at line 515 of file types.h.

### 6.5.2.22 VTSS\_PRIO\_NO\_NONE

```
#define VTSS_PRIO_NO_NONE 0xffffffff
```

Priority number none (= undefined)

Definition at line 516 of file types.h.

### 6.5.2.23 VTSS\_PRIO\_START

```
#define VTSS_PRIO_START 0
```

Priority start number (lowest)

Definition at line 517 of file types.h.

### 6.5.2.24 VTSS\_PRIO\_END

```
#define VTSS_PRIO_END (VTSS_PRIO_START + VTSS_PRIOS)
```

Priority end number

Definition at line 518 of file types.h.

### 6.5.2.25 VTSS\_PRIO\_ARRAY\_SIZE

```
#define VTSS_PRIO_ARRAY_SIZE VTSS_PRIO_END
```

Priority number array size

Definition at line 519 of file types.h.

### 6.5.2.26 VTSS\_QUEUES

```
#define VTSS_QUEUES VTSS_PRIOS
```

Number of queues

Definition at line 523 of file types.h.

### 6.5.2.27 VTSS\_QUEUE\_START

```
#define VTSS_QUEUE_START 0
```

Queue start number

Definition at line 524 of file types.h.

### 6.5.2.28 VTSS\_QUEUE\_END

```
#define VTSS_QUEUE_END (VTSS_QUEUE_START + VTSS_QUEUES)
```

Queue end number

Definition at line 525 of file types.h.

### 6.5.2.29 VTSS\_QUEUE\_ARRAY\_SIZE

```
#define VTSS_QUEUE_ARRAY_SIZE VTSS_QUEUE_END
```

Queue number array size

Definition at line 526 of file types.h.

### 6.5.2.30 VTSS\_PCPS

```
#define VTSS_PCPS 8
```

Number of PCP values

Definition at line 530 of file types.h.

### 6.5.2.31 VTSS\_PCP\_START

```
#define VTSS_PCP_START 0
```

PCP start number

Definition at line 531 of file types.h.

### 6.5.2.32 VTSS\_PCP\_END

```
#define VTSS_PCP_END (VTSS_PCP_START + VTSS_PCPS)
```

PCP end number

Definition at line 532 of file types.h.

### 6.5.2.33 VTSS\_PCP\_ARRAY\_SIZE

```
#define VTSS_PCP_ARRAY_SIZE VTSS_PCP_END
```

PCP array size

Definition at line 533 of file types.h.

### 6.5.2.34 VTSS\_DEIS

```
#define VTSS_DEIS 2
```

Number of DEI values

Definition at line 537 of file types.h.

### 6.5.2.35 VTSS\_DEI\_START

```
#define VTSS_DEI_START 0
```

DEI start number

Definition at line 538 of file types.h.

### 6.5.2.36 VTSS\_DEI\_END

```
#define VTSS_DEI_END (VTSS_DEI_START + VTSS_DEIS)
```

DEI end number

Definition at line 539 of file types.h.

### 6.5.2.37 VTSS\_DEI\_ARRAY\_SIZE

```
#define VTSS_DEI_ARRAY_SIZE VTSS_DEI_END
```

DEI array size

Definition at line 540 of file types.h.

**6.5.2.38 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.

**6.5.2.39 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.

**6.5.2.40 VTSS\_DPL\_START**

```
#define VTSS_DPL_START 0
```

DPL start number

Definition at line 551 of file types.h.

**6.5.2.41 VTSS\_DPL\_END**

```
#define VTSS_DPL_END (VTSS_DPL_START + VTSS_DPLS)
```

DPL end number

Definition at line 552 of file types.h.

**6.5.2.42 VTSS\_DPL\_ARRAY\_SIZE**

```
#define VTSS_DPL_ARRAY_SIZE VTSS_DPL_END
```

DPL array size

Definition at line 553 of file types.h.

#### 6.5.2.43 VTSS\_BITRATE\_DISABLED

```
#define VTSS_BITRATE_DISABLED 0xffffffff
```

Bitrate disabled

Definition at line 563 of file types.h.

#### 6.5.2.44 VTSS\_VID\_NULL

```
#define VTSS_VID_NULL ((const vtss_vid_t)0)
```

NULL VLAN ID

Definition at line 609 of file types.h.

#### 6.5.2.45 VTSS\_VID\_DEFAULT

```
#define VTSS_VID_DEFAULT ((const vtss_vid_t)1)
```

Default VLAN ID

Definition at line 610 of file types.h.

#### 6.5.2.46 VTSS\_VID\_RESERVED

```
#define VTSS_VID_RESERVED ((const vtss_vid_t)0xFFFF)
```

Reserved VLAN ID

Definition at line 611 of file types.h.

#### 6.5.2.47 VTSS\_VIDS

```
#define VTSS_VIDS ((const vtss_vid_t)4096)
```

Number of VLAN IDs

Definition at line 612 of file types.h.

### 6.5.2.48 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.

### 6.5.2.49 VTSS\_ETYPE\_VTSS

```
#define VTSS_ETYPE_VTSS 0x8880
```

Vitesse Ethernet Type

Definition at line 640 of file types.h.

### 6.5.2.50 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.

### 6.5.2.51 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.

### 6.5.2.52 VTSS\_EVCS

```
#define VTSS_EVCS 256
```

Maximum number of Ethernet Virtual Connections

Definition at line 670 of file types.h.

### 6.5.2.53 VTSS\_ISDX\_NONE

```
#define VTSS_ISDX_NONE (0)
```

Ingress Service Index number none

Definition at line 674 of file types.h.

### 6.5.2.54 VTSS\_AGGRS

```
#define VTSS_AGGRS (VTSS_PORTS/2)
```

Number of LLAGs

Definition at line 678 of file types.h.

### 6.5.2.55 VTSS\_AGGR\_NO\_NONE

```
#define VTSS_AGGR_NO_NONE 0xffffffff
```

Aggregation number none

Definition at line 679 of file types.h.

### 6.5.2.56 VTSS\_AGGR\_NO\_START

```
#define VTSS_AGGR_NO_START 0
```

Aggregation start number

Definition at line 680 of file types.h.

### 6.5.2.57 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.

### 6.5.2.58 VTSS\_GLAGS

```
#define VTSS_GLAGS 2
```

Number of GLAGs

Definition at line 689 of file types.h.

### 6.5.2.59 VTSS\_GLAG\_NO\_NONE

```
#define VTSS_GLAG_NO_NONE 0xffffffff
```

GLAG number none

Definition at line 691 of file types.h.

### 6.5.2.60 VTSS\_GLAG\_NO\_START

```
#define VTSS_GLAG_NO_START 0
```

GLAG start number

Definition at line 692 of file types.h.

### 6.5.2.61 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.

### 6.5.2.62 VTSS\_GLAG\_PORTS

```
#define VTSS_GLAG_PORTS 8
```

Number of GLAG ports

Definition at line 696 of file types.h.

### 6.5.2.63 VTSS\_GLAG\_PORT\_START

```
#define VTSS_GLAG_PORT_START 0
```

GLAG port start number

Definition at line 697 of file types.h.

### 6.5.2.64 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.

### 6.5.2.65 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.

### 6.5.2.66 VTSS\_PACKET\_RX\_QUEUE\_CNT

```
#define VTSS_PACKET_RX_QUEUE_CNT 8
```

Number of Rx packet queues

Definition at line 729 of file types.h.

### 6.5.2.67 VTSS\_PACKET\_RX\_GRP\_CNT

```
#define VTSS_PACKET_RX_GRP_CNT 2
```

Number of Rx packet groups to which any queue can map

Definition at line 731 of file types.h.

### 6.5.2.68 VTSS\_PACKET\_TX\_GRP\_CNT

```
#define VTSS_PACKET_TX_GRP_CNT 2
```

Number of Tx packet groups

Definition at line 733 of file types.h.

### 6.5.2.69 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.

### 6.5.2.70 VTSS\_PACKET\_RX\_QUEUE\_START

```
#define VTSS_PACKET_RX_QUEUE_START (0)
```

Rx queue start number

Definition at line 746 of file types.h.

### 6.5.2.71 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.

### 6.5.2.72 VTSS\_ACL\_POLICERS

```
#define VTSS_ACL_POLICERS 16
```

Number of ACL policers

Definition at line 1028 of file types.h.

### 6.5.2.73 VTSS\_ACL\_POLICER\_NO\_START

```
#define VTSS_ACL_POLICER_NO_START 0
```

ACL policer start number

Definition at line 1029 of file types.h.

### 6.5.2.74 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.

### 6.5.2.75 VTSS\_ACL\_POLICY\_NO\_NONE

```
#define VTSS_ACL_POLICY_NO_NONE 0xffffffff
```

ACLs disabled on port

Definition at line 1034 of file types.h.

### 6.5.2.76 VTSS\_ACL\_POLICY\_NO\_MIN

```
#define VTSS_ACL_POLICY_NO_MIN 0
```

ACLs policy minimum number

Definition at line 1035 of file types.h.

### 6.5.2.77 VTSS\_ACL\_POLICY\_NO\_MAX

```
#define VTSS_ACL_POLICY_NO_MAX 255
```

ACLs policy maximum number

Definition at line 1037 of file types.h.

### 6.5.2.78 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.

### 6.5.2.79 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.

### 6.5.2.80 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.

### 6.5.2.81 VTSS\_HQOS\_COUNT

```
#define VTSS_HQOS_COUNT 256
```

Maximum number of HQoS entries

Definition at line 1165 of file types.h.

### 6.5.2.82 VTSS\_HQOS\_ID\_NONE

```
#define VTSS_HQOS_ID_NONE 0xfffff
```

Special HQoS ID value

Definition at line 1167 of file types.h.

### 6.5.2.83 VTSS\_ONE\_MIA

```
#define VTSS_ONE_MIA 1000000000
```

One billion

Definition at line 1198 of file types.h.

### 6.5.2.84 VTSS\_ONE\_MILL

```
#define VTSS_ONE_MILL 1000000
```

One million

Definition at line 1199 of file types.h.

### 6.5.2.85 VTSS\_MAX\_TIMEINTERVAL

```
#define VTSS_MAX_TIMEINTERVAL 0x7fffffffffffffLL
```

Maximum time interval

Definition at line 1200 of file types.h.

### 6.5.2.86 VTSS\_INTERVAL\_SEC

```
#define VTSS_INTERVAL_SEC( t ) ((i32)VTSS_DIV64((t)>>16, VTSS_ONE_MIA))
```

One Second time interval

Definition at line 1202 of file types.h.

### 6.5.2.87 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.

### 6.5.2.88 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.

### 6.5.2.89 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.

### 6.5.2.90 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.

### 6.5.2.91 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.

### 6.5.2.92 VTSS\_CLOCK\_IDENTITY\_LENGTH

```
#define VTSS_CLOCK_IDENTITY_LENGTH 8
```

Length of unique PTP identifier

Definition at line 1218 of file types.h.

### 6.5.2.93 VTSS\_SYNCE\_CLK\_PORT\_ARRAY\_SIZE

```
#define VTSS_SYNCE_CLK_PORT_ARRAY_SIZE 4
```

SYNCE clock out port numberarray size

Definition at line 1230 of file types.h.

## 6.5.3 Typedef Documentation

### 6.5.3.1 i8

```
typedef char i8
```

Fallback Integer types.

8-bit signed

Definition at line 68 of file types.h.

### 6.5.3.2 i16

```
typedef signed short i16
```

16-bit signed

Definition at line 69 of file types.h.

### 6.5.3.3 i32

```
typedef signed int i32
```

32-bit signed

Definition at line 70 of file types.h.

### 6.5.3.4 i64

```
typedef signed long long i64
```

64-bit signed

Definition at line 71 of file types.h.

### 6.5.3.5 u8

```
typedef unsigned char u8
```

8-bit unsigned

Definition at line 73 of file types.h.

### 6.5.3.6 u16

```
typedef unsigned short u16
```

16-bit unsigned

Definition at line 74 of file types.h.

### 6.5.3.7 u32

```
typedef unsigned int u32
```

32-bit unsigned

Definition at line 75 of file types.h.

### 6.5.3.8 u64

```
typedef unsigned long long u64
```

64-bit unsigned

Definition at line 76 of file types.h.

### 6.5.3.9 BOOL

```
typedef unsigned char BOOL
```

Boolean implemented as 8-bit unsigned

Definition at line 78 of file types.h.

**6.5.3.10 uintptr\_t**

```
typedef unsigned int uintptr_t
```

Unsigned integer big enough to hold pointers

Definition at line 79 of file types.h.

**6.5.3.11 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.

**6.5.3.12 vtss\_isdx\_t**

```
typedef u32 vtss_isdx_t
```

Ingress Service Index type

Definition at line 673 of file types.h.

**6.5.3.13 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.

**6.5.3.14 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.

## 6.5.4 Enumeration Type Documentation

**6.5.4.1 anonymous enum**

```
anonymous enum
```

Error codes.

## Enumerator

VTSS_RC_OK	Success
VTSS_RC_ERROR	Unspecified error
VTSS_RC_INV_STATE	Invalid state for operation
VTSS_RC_INCOMPLETE	Incomplete result
VTSS_RC_ERR_CLK_CONF_NOT_SUPPORTED	The PHY doesn't support clock configuration (for SyncE)
VTSS_RC_ERR_KR_CONF_NOT_SUPPORTED	The PHY doesn't support 10GBASE_KR equalization
VTSS_RC_ERR_KR_CONF_INVALID_PARAMETER	One of the parameters are out of range
VTSS_RC_ERR_PHY_BASE_NO_NOT_FOUND	Port base number (first port within a chip) is not found
VTSS_RC_ERR_PHY_6G_MACRO_SETUP	Setup of 6G macro failed
VTSS_RC_ERR_PHY_MEDIA_IF_NOT_SUPPORTED	PHY does not support the selected media mode
VTSS_RC_ERR_PHY_CLK_CONF_NOT_SUPPORTED	The PHY doesn't support clock configuration (for SyncE)
VTSS_RC_ERR_PHY_GPIO_ALT_MODE_NOT_SUPPORTED	The PHY doesn't support the alternative mode for the selected GPIO pin
VTSS_RC_ERR_PHY_GPIO_PIN_NOT_SUPPORTED	The PHY doesn't support the selected GPIO pin
VTSS_RC_ERR_PHY_PORT_OUT_RANGE	PHY API called with port number larger than VTSS_PORTS
VTSS_RC_ERR_PHY_PATCH_SETTING_NOT_SUPPORTED	PHY API micro patch setting not supported for the port in question
VTSS_RC_ERR_PHY_LCPLL_NOT_SUPPORTED	PHY API LC-PLL status not supported for the port
VTSS_RC_ERR_PHY_RCPLL_NOT_SUPPORTED	PHY API RC-PLL status not supported for the port
VTSS_RC_ERR_MACSEC_INVALID_SCI_MACADDR	From IEEE 802.1AE-2006, section 9.9 - The 64-bit value FF-FF-FF-FF-FF-FF is never used as an SCI and is reserved for use by implementations to indicate the absence of an SC or an SCI in contexts where an SC can be present
VTSS_RC_ERR_MACSEC_NOT_ENABLED	Trying to access port where MACSEC is not enabled
VTSS_RC_ERR_MACSEC_SECY_ALREADY_IN_USE	Trying to use a secy which is already in use
VTSS_RC_ERR_MACSEC_NO_SECY_FOUND	No SecY found for the specific port
VTSS_RC_ERR_MACSEC_NO_SECY_VACANCY	No secy vacancy
VTSS_RC_ERR_MACSEC_INVALID_VALIDATE_FRAME	Validate_frames value invalid
VTSS_RC_ERR_MACSEC_COULD_NOT_PRG_SA_MATCH	Could not program the SA match
VTSS_RC_ERR_MACSEC_COULD_NOT_PRG_SA_FLOW	Could not program the SA flow
VTSS_RC_ERR_MACSEC_COULD_NOT_ENA_SA	Could not enable the SA
VTSS_RC_ERR_MACSEC_COULD_NOT_SET_SA	Could not set SA to in use
VTSS_RC_ERR_MACSEC_INVALID_BYPASS_HEADER_LENGTH	Invalid header bypass length
VTSS_RC_ERR_MACSEC_SC_NOT_FOUND	Could not find SC (from sci)
VTSS_RC_ERR_MACSEC_NO_CTRL_FRAME_MATCH	No control frame match
VTSS_RC_ERR_MACSEC_COULD_NOT_SET_PATTERN	Could no set bypass pattern for CP rule
VTSS_RC_ERR_MACSEC_TIMEOUT_ISSUE	Internal timeout issue, bailing out

## Enumerator

VTSS_RC_ERR_MACSEC_COULD_NOT_EMPT← Y_EGRESS	Could not empty the egress pipeline
VTSS_RC_ERR_MACSEC_AN_NOT_CREATED	AN not created.
VTSS_RC_ERR_MACSEC_COULD_NOT_EMPT← Y_INGRESS	Could not empty the ingress pipeline
VTSS_RC_ERR_MACSEC_TX_SC_NOT_EXIST	No tx SC found
VTSS_RC_ERR_MACSEC_COULD_NOT_DISAB← LE_SA	Could not disable sa
VTSS_RC_ERR_MACSEC_COULD_NOT_DEL_R← X_SA	Could not delete rx sa
VTSS_RC_ERR_MACSEC_COULD_NOT_DEL_T← X_SA	Could not delete tx sa
VTSS_RC_ERR_MACSEC_PATTERN_NOT_SET	Pattern not set
VTSS_RC_ERR_MACSEC_HW_RESOURCE_EX← HUSTED	HW resources exhausted
VTSS_RC_ERR_MACSEC_SCI_ALREADY_EXISTS	SCI already exists
VTSS_RC_ERR_MACSEC_SC_RESOURCE_NO← T_FOUND	Could not find SC resources
VTSS_RC_ERR_MACSEC_RX_AN_ALREADY_I← N_USE	Rx AN is in use
VTSS_RC_ERR_MACSEC_EMPTY_RECORD	Could not get an empty record
VTSS_RC_ERR_MACSEC_COULD_NOT_PRG← XFORM	Could not program the xform record
VTSS_RC_ERR_MACSEC_COULD_NOT_TOGG← LE_SA	Could not toggle SA
VTSS_RC_ERR_MACSEC_TX_AN_ALREADY_I← N_USE	Tx AN is in use
VTSS_RC_ERR_MACSEC_ALL_AVAILABLE_SA← _IN_USE	All available SA's are in use
VTSS_RC_ERR_MACSEC_MATCH_DISABLE	MACSEC match disabled
VTSS_RC_ERR_MACSEC_ALL_CP_RULES_IN← USE	All CP rules of the specific type are in use
VTSS_RC_ERR_MACSEC_PATTERN_PRIO_NO← T_VALID	The pattern priority is not valid
VTSS_RC_ERR_MACSEC_BUFFER_TOO_SMALL	Buffer to small, must be greater than VTSS_MACSEC_FRAME_CAPTURE_SIZE_MAX
VTSS_RC_ERR_MACSEC_FRAME_TOO_LONG	Frame length is supposed to be less than the amount of data in the fifo
VTSS_RC_ERR_MACSEC_FRAME_TRUNCATED	Frame is Truncated
VTSS_RC_ERR_MACSEC_PHY_POWERED_DO← WN	Phy is powered down, i.e. the MacSec block is not accessible
VTSS_RC_ERR_MACSEC_PHY_NOT_MACSEC← _CAPABLE	Port/Phy is not MacSec capable
VTSS_RC_ERR_MACSEC_AN_NOT_EXIST	AN does not exist
VTSS_RC_ERR_MACSEC_NO_PATTERN_CFG	No pattern is configured
VTSS_RC_ERR_MACSEC_MAX_MTU	Maximum MTU allowed is 32761 (+ 4 bytes for VLAN)
VTSS_RC_ERR_MACSEC_UNEXPECT_CP_MODE	Unexpected CP mode
VTSS_RC_ERR_MACSEC_COULD_NOT_DISAB← LE_AN	Could not disable AN
VTSS_RC_ERR_MACSEC_RULE_OUT_OF_RAN← GE	Rule id is out of range. Must not be larger than VTSS_MACSEC_CP_RULES

**Enumerator**

VTSS_RC_ERR_MACSEC_RULE_NOT_EXIST	Rule does not exist
VTSS_RC_ERR_MACSEC_CSR_READ	Could not do CSR read
VTSS_RC_ERR_MACSEC_CSR_WRITE	Could not do CSR write
VTSS_RC_ERR_PHY_6G_RCPLL_ON_BASE_PORT_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.

**6.5.4.2 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.

**6.5.4.3 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.

## 6.5.4.4 vtss\_serdes\_mode\_t

```
enum vtss_serdes_mode_t
```

Serdess 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.

#### 6.5.4.5 `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.

#### 6.5.4.6 `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.

#### 6.5.4.7 `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.

#### 6.5.4.8 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.

#### 6.5.4.9 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.

#### 6.5.4.10 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.

#### 6.5.4.11 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.

#### 6.5.4.12 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.

#### 6.5.4.13 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.

## 6.5.4.14 vtss\_hqos\_sch\_mode\_t

```
enum vtss_hqos_sch_mode_t
```

HQoS port scheduling mode.

The scheduling mode for the port affects which egress QoS options are available.

Enumerator

VTSS_HQOS_SCH_MODE_NORMAL	Normal QoS configuration available for non-service traffic only (default)
VTSS_HQOS_SCH_MODE_BASIC	Basic QoS configuration available for non-service traffic only
VTSS_HQOS_SCH_MODE_HIERARCHICAL	Basic QoS configuration available per HQoS entry (HQoS)

Definition at line 1173 of file types.h.

## 6.6 vtss\_api/include/vtss\_ae\_api.h File Reference

ae API

```
#include <vtss/api/types.h>
```

### 6.6.1 Detailed Description

ae API

## 6.7 vtss\_api/include/vtss\_afi\_api.h File Reference

AFI API.

```
#include <vtss/api/options.h>
```

### 6.7.1 Detailed Description

AFI API.

This header file describes Automatic Frame Injector functions.

## 6.8 vtss\_api/include/vtss\_aneg\_api.h File Reference

ANEG API.

```
#include <vtss/api/types.h>
```

### 6.8.1 Detailed Description

ANEG API.

## 6.9 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_fdma_api.h>
#include <vtss_sync_api.h>
#include <vtss_phy_ts_api.h>
#include <vtss_ts_api.h>
#include <vtss_wis_api.h>
```

### 6.9.1 Detailed Description

Vitesse API main header file.

This is the only header file which must be included by the application

## 6.10 vtss\_api/include/vtss\_evc\_api.h File Reference

EVC API.

```
#include <vtss/api/types.h>
```

### Typedefs

- **typedef vtss\_dlb\_policer\_conf\_t vtss\_evc\_policer\_conf\_t**  
*EVC policer configuration.*

## Functions

- `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.*

### 6.10.1 Detailed Description

EVC API.

This header file describes EVC functions

### 6.10.2 Function Documentation

#### 6.10.2.1 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

<code>inst</code>	[IN] Target instance reference.
<code>policer_id</code>	[IN] Policer ID.
<code>conf</code>	[OUT] Policer configuration.

#### Returns

Return code.

#### 6.10.2.2 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.

## 6.11 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 10
- #define VTSS\_FDMA\_DCB\_SIZE\_BYTES 16
- #define VTSS\_FDMA\_HDR\_SIZE\_BYTES 32
- #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)

## 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\_ch\_usage\_t { VTSS\_FDMA\_CH\_USAGE\_UNUSED, VTSS\_FDMA\_CH\_USAGE\_XTR, VTSS\_FDMA\_CH\_USAGE\_INJ }**  
*Channel usage.*
- **enum vtss\_fdma\_dcb\_type\_t { VTSS\_FDMA\_DCB\_TYPE\_XTR, VTSS\_FDMA\_DCB\_TYPE\_INJ }**  
*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\_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.*

### 6.11.1 Detailed Description

Frame DMA API.

### 6.11.2 Macro Definition Documentation

### 6.11.2.1 VTSS\_FDMA\_CH\_CNT

```
#define VTSS_FDMA_CH_CNT 10
```

Number of DMA Channels. Fixed layout

Definition at line 222 of file vtss\_fdma\_api.h.

### 6.11.2.2 VTSS\_FDMA\_DCB\_SIZE\_BYTES

```
#define VTSS_FDMA_DCB_SIZE_BYTES 16
```

Number of bytes in a DCB.

Definition at line 223 of file vtss\_fdma\_api.h.

### 6.11.2.3 VTSS\_FDMA\_HDR\_SIZE\_BYTES

```
#define VTSS_FDMA_HDR_SIZE_BYTES 32
```

Max(XTR\_HDR\_SZ, INJ\_HDR\_SZ). Worst-case is INJ (28 for IFH + 4 for VLAN tag (if @switch\_frm == TRUE)

Definition at line 224 of file vtss\_fdma\_api.h.

### 6.11.2.4 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 299 of file vtss\_fdma\_api.h.

### 6.11.2.5 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 302 of file vtss\_fdma\_api.h.

### 6.11.2.6 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 306 of file vtss\_fdma\_api.h.

### 6.11.2.7 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 307 of file vtss\_fdma\_api.h.

### 6.11.2.8 VTSS\_FDMA\_MIN\_DATA\_PER\_NON\_SOFR\_DCB\_BYTES

```
#define VTSS_FDMA_MIN_DATA_PER_NON_SOFR_DCB_BYTES 1
```

Defines the minimum size in bytes of an injection/extraction non-SOF- DCB's associated data area.

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

### 6.11.2.9 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 314 of file vtss\_fdma\_api.h.

### 6.11.2.10 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 317 of file vtss\_fdma\_api.h.

#### 6.11.2.11 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 are e.g. placing variables on cache-line boundaries.

Definition at line 321 of file vtss\_fdma\_api.h.

#### 6.11.2.12 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 929 of file vtss\_fdma\_api.h.

### 6.11.3 Typedef Documentation

### 6.11.3.1 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 CPI Is does the EDMA 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.

**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 disjoint.

**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`.

**6.11.3.2 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.

**6.11.4 Enumeration Type Documentation****6.11.4.1 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.

Definition at line 1547 of file vtss\_fdma\_api.h.

## 6.11.4.2 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> .

Definition at line 2580 of file vtss\_fdma\_api.h.

## 6.11.5 Function Documentation

## 6.11.5.1 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.

**6.11.5.2 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.

**6.11.5.3 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.

**6.11.5.4 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

**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.

**6.11.5.5 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.

**6.11.5.6 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.

**6.11.5.7 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.

**6.11.5.8 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.

**6.11.5.9 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.

**6.11.5.10 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.

**6.11.5.11 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. IRQs 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 should 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>ctxt</i>	[IN]: A user-defined argument, which is passed to the <code>inj_post_cb()</code> and <code>xtr_cb()</code> callback functions. Rarely needed.

#### Returns

`VTSS_RC_OK` on success.

`VTSS_RC_ERROR` if @inst parameter was erroneous.

## 6.12 vtss\_api/include/vtss\_gfp\_api.h File Reference

GFP API.

```
#include <vtss/api/types.h>
```

### 6.12.1 Detailed Description

GFP API.

## 6.13 vtss\_api/include/vtss\_hqos\_api.h File Reference

HQoS API.

```
#include <vtss/api/types.h>
```

### 6.13.1 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.

## 6.14 vtss\_api/include/vtss\_i2c\_api.h File Reference

I2C API.

```
#include <vtss/api/types.h>
```

### 6.14.1 Detailed Description

I2C API.

## 6.15 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\\_init\\_conf\\_t](#)  
*Initialization configuration.*
- struct [vtss\\_restart\\_status\\_t](#)  
*Restart status.*

## Macros

- #define VTSS\_I2C\_NO\_MULTIPLEXER -1

## Typedefs

- typedef `vtss_rc(* vtss_reg_read_t)` (const `vtss_chip_no_t` chip\_no, const `u32` addr, `u32 *const value`)  
*Register read function.*
- typedef `vtss_rc(* vtss_reg_write_t)` (const `vtss_chip_no_t` chip\_no, const `u32` addr, const `u32` value)  
*Register write function.*
- typedef `vtss_rc(* vtss_i2c_read_t)` (const `vtss_port_no_t` port\_no, const `u8` i2c\_addr, const `u8` addr, `u8 *const data`, const `u8` cnt, const `i8` i2c\_clk\_sel)  
*I2C read function.*
- typedef `vtss_rc(* vtss_i2c_write_t)` (const `vtss_port_no_t` port\_no, const `u8` i2c\_addr, `u8 *const data`, const `u8` cnt, const `i8` i2c\_clk\_sel)  
*I2C write function.*
- typedef `vtss_rc(* vtss_spi_read_write_t)` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `u8` bit-size, `u8 *const bitstream`)  
*SPI read/write function.*
- typedef `vtss_rc(* vtss_spi_32bit_read_write_t)` (const `vtss_inst_t` inst, `vtss_port_no_t` port\_no, `BOOL` read, `u8` dev, `u16` reg\_num, `u32 *const data`)  
*SPI 32 bit read/write function.*
- typedef `vtss_rc(* vtss_spi_64bit_read_write_t)` (const `vtss_inst_t` inst, `vtss_port_no_t` port\_no, `BOOL` read, `u8` dev, `u16` reg\_num, `u64 *const data`)  
*SPI 64 bit read/write function.*
- typedef `vtss_rc(* vtss_miim_read_t)` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `u8` addr, `u16 *const value`)  
*MII management read function (IEEE 802.3 clause 22)*
- typedef `vtss_rc(* vtss_miim_write_t)` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `u8` addr, const `u16` value)  
*MII management write function (IEEE 802.3 clause 22)*
- typedef `vtss_rc(* vtss_mmd_read_t)` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `u8` mmd, const `u16` addr, `u16 *const value`)  
*MMD management read function (IEEE 802.3 clause 45)*
- typedef `vtss_rc(* vtss_mmd_read_inc_t)` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `u8` mmd, const `u16` addr, `u16 *const buf, u8 count`)  
*MMD management read increment function (IEEE 802.3 clause 45)*
- typedef `vtss_rc(* vtss_mmd_write_t)` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `u8` mmd, const `u16` addr, const `u16` value)  
*MMD management write function (IEEE 802.3 clause 45)*
- typedef `u16 vtss_version_t`  
*API version.*

## Enumerations

- enum `vtss_target_type_t` {
 `VTSS_TARGET_CU_PHY`, `VTSS_TARGET_10G_PHY`, `VTSS_TARGET_SPARX_III_11` = 0x7414,  
`VTSS_TARGET_SERVAL_LITE` = 0x7416,  
`VTSS_TARGET_SERVAL` = 0x7418, `VTSS_TARGET_SEVILLE` = 0x9953, `VTSS_TARGET_SPARX_III_10 UM`  
= 0x7420, `VTSS_TARGET_SPARX_III_17 UM` = 0x7421,  
`VTSS_TARGET_SPARX_III_25 UM` = 0x7422, `VTSS_TARGET_CARACAL_LITE` = 0x7423, `VTSS_TARGET_SPARX_III_10`  
= 0x7424, `VTSS_TARGET_SPARX_III_18` = 0x7425,
 }

```

VTSS_TARGET_SPARX_III_24 = 0x7426, VTSS_TARGET_SPARX_III_26 = 0x7427, VTSS_TARGET_SPARX_III_10_01
= 0x17424, VTSS_TARGET_CARACAL_1 = 0x7428,
VTSS_TARGET_CARACAL_2 = 0x7429, VTSS_TARGET_JAGUAR_1 = 0x7460, VTSS_TARGET_LYNX_1
= 0x7462, VTSS_TARGET_E_STAX_III_48 = 0x7432,
VTSS_TARGET_E_STAX_III_68 = 0x7434, VTSS_TARGET_E_STAX_III_24_DUAL = 0xD7431,
VTSS_TARGET_E_STAX_III_68_DUAL = 0xD7434, VTSS_TARGET_DAYTONA = 0x8492,
VTSS_TARGET_TALLADEGA = 0x8494, VTSS_TARGET_SERVAL_2 = 0x7438, VTSS_TARGET_LYNX_2
= 0x7464, VTSS_TARGET_JAGUAR_2 = 0x7468,
VTSS_TARGET_SPARX_IV_52 = 0x7442, VTSS_TARGET_SPARX_IV_44 = 0x7444, VTSS_TARGET_SPARX_IV_80
= 0x7448, VTSS_TARGET_SPARX_IV_90 = 0x7449 }

```

*Target chip type.*

- enum `vtss_pi_width_t` { `VTSS_PI_WIDTH_16` = 0, `VTSS_PI_WIDTH_8` }  
*PI data width.*
- enum `vtss_port_mux_mode_t` { `VTSS_PORT_MUX_MODE_AUTO`, `VTSS_PORT_MUX_MODE_0`,
`VTSS_PORT_MUX_MODE_1`, `VTSS_PORT_MUX_MODE_2` }  
*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_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)*

### 6.15.1 Detailed Description

Initialization API.

This header file describes functions used to create and initialize targets

## 6.15.2 Macro Definition Documentation

### 6.15.2.1 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.

## 6.15.3 Typedef Documentation

### 6.15.3.1 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.

### 6.15.3.2 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.

**6.15.3.3 vtss\_i2c\_read\_t**

```
typedef vtss_rc(* vtss_i2c_read_t) (const vtss_port_no_t port_no, const u8 i2c_addr, const u8
addr, u8 *const data, const u8 cnt, const i8 i2c_clk_sel)
```

I2C read function.

**Parameters**

<i>port_no</i>	[IN] Port number
<i>i2c_addr</i>	[IN] I2C device address
<i>addr</i>	[IN] Register address
<i>data</i>	[OUT] Pointer the register(s) data value.
<i>cnt</i>	[IN] Number of registers to read
<i>i2c_clk_sel</i>	[IN] If i2c clock multiplexing is supported then this is the i2c mux, else use VTSS_I2C_NO_MULTIPLEXER

**Returns**

Return code.

Definition at line 152 of file vtss\_init\_api.h.

**6.15.3.4 vtss\_i2c\_write\_t**

```
typedef vtss_rc(* vtss_i2c_write_t) (const vtss_port_no_t port_no, const u8 i2c_addr, u8 *const
data, const u8 cnt, const i8 i2c_clk_sel)
```

I2C write function.

**Parameters**

<i>port_no</i>	[IN] Port number
<i>i2c_addr</i>	[IN] I2C device address
<i>data</i>	[OUT] Pointer the data to be written.
<i>cnt</i>	[IN] Number of data bytes to write
<i>i2c_clk_sel</i>	[IN] If i2c clock multiplexing is supported then this is the i2c mux, else use VTSS_I2C_NO_MULTIPLEXER

**Returns**

Return code.

Definition at line 170 of file vtss\_init\_api.h.

**6.15.3.5 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.

**6.15.3.6 vtss\_spi\_32bit\_read\_write\_t**

```
typedef vtss_rc(* vtss_spi_32bit_read_write_t) (const vtss_inst_t inst, vtss_port_no_t port_no, BOOL read, u8 dev, u16 reg_num, u32 *const data)
```

SPI 32 bit read/write function.

**Parameters**

<i>inst</i>	[IN] Vitesse API instance.
<i>port_no</i>	[IN] Port number.
<i>read</i>	[IN] Read/Write.
<i>dev</i>	[IN] MMD device number.
<i>reg_num</i>	[IN] Register offset.
<i>data</i>	[IN OUT] Pointer to the data to be written to SPI Slave, if doing write operation. Pointer to the data read from SPI Slave, if doing read operation.

**Returns**

Return code.

Definition at line 205 of file vtss\_init\_api.h.

**6.15.3.7 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.

**6.15.3.8 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.

**6.15.3.9 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.

**6.15.3.10 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.

#### 6.15.3.11 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.

#### 6.15.3.12 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.

## 6.15.4 Enumeration Type Documentation

### 6.15.4.1 `vtss_target_type_t`

`enum vtss_target_type_t`

Target chip type.

#### Enumerator

<code>VTSS_TARGET CU PHY</code>	Cu PHY family
<code>VTSS_TARGET_10G_PHY</code>	10G PHY family
<code>VTSS_TARGET_SPARX_III_11</code>	SparX-III-11 SME switch
<code>VTSS_TARGET_SERVAL_LITE</code>	Serval Lite CE switch
<code>VTSS_TARGET_SERVAL</code>	Serval CE switch
<code>VTSS_TARGET_SEVILLE</code>	Seville switch
<code>VTSS_TARGET_SPARX_III_10_UM</code>	SparxIII-10 unmanaged switch
<code>VTSS_TARGET_SPARX_III_17_UM</code>	SparxIII-17 unmanaged switch
<code>VTSS_TARGET_SPARX_III_25_UM</code>	SparxIII-25 unmanaged switch
<code>VTSS_TARGET_CARACAL_LITE</code>	Caracal-Lite CE switch
<code>VTSS_TARGET_SPARX_III_10</code>	SparxIII-10 switch
<code>VTSS_TARGET_SPARX_III_18</code>	SparxIII-18 switch
<code>VTSS_TARGET_SPARX_III_24</code>	SparxIII-24 switch
<code>VTSS_TARGET_SPARX_III_26</code>	SparxIII-26 switch
<code>VTSS_TARGET_SPARX_III_10_01</code>	SparxIII-10-01 switch
<code>VTSS_TARGET_CARACAL_1</code>	Caracal-1 CE switch
<code>VTSS_TARGET_CARACAL_2</code>	Caracal-2 CE switch
<code>VTSS_TARGET_JAGUAR_1</code>	Jaguar-1 CE switch
<code>VTSS_TARGET_LYNX_1</code>	LynX-1 CE switch
<code>VTSS_TARGET_E_STAX_III_48</code>	E-StaX-III-48
<code>VTSS_TARGET_E_STAX_III_68</code>	E-StaX-III-68
<code>VTSS_TARGET_E_STAX_III_24_DUAL</code>	Dual E-StaX-III-24
<code>VTSS_TARGET_E_STAX_III_68_DUAL</code>	Dual E-StaX-III-68
<code>VTSS_TARGET_DAYTONA</code>	Daytona FEC OTN Phy
<code>VTSS_TARGET_TALLADEGA</code>	Talladega FEC OTN Phy
<code>VTSS_TARGET_SERVAL_2</code>	Serval-2 CE switch
<code>VTSS_TARGET_LYNX_2</code>	LynX-2 CE switch
<code>VTSS_TARGET_JAGUAR_2</code>	Jaguar-2 CE switch
<code>VTSS_TARGET_SPARX_IV_52</code>	Sparx-IV-52 switch
<code>VTSS_TARGET_SPARX_IV_44</code>	Sparx-IV-44 switch
<code>VTSS_TARGET_SPARX_IV_80</code>	Sparx-IV-80 switch
<code>VTSS_TARGET_SPARX_IV_90</code>	Sparx-IV-80 switch

Definition at line 42 of file `vtss_init_api.h`.

#### 6.15.4.2 vtss\_port\_mux\_mode\_t

enum `vtss_port_mux_mode_t`

Port mux configuration.

Enumerator

<code>VTSS_PORT_MUX_MODE_AUTO</code>	Port mux mode autodetected (not possible for speeds close to 80Gb (aggregated))
<code>VTSS_PORT_MUX_MODE_0</code>	Ports muxed to Serdes blocks: 24x2G5, 4x10Gb, NPI, chip ports 8-23,48-53
<code>VTSS_PORT_MUX_MODE_1</code>	Ports muxed to Serdes blocks: 32x2G5, 2x10Gb, NPI, chip ports 8-31,48-50
<code>VTSS_PORT_MUX_MODE_2</code>	Supports 48x1G + 4x10G + NPI (JR2-RevC)

Definition at line 335 of file vtss\_init\_api.h.

#### 6.15.4.3 vtss\_restart\_t

enum `vtss_restart_t`

Restart type.

Enumerator

<code>VTSS_RESTART_COLD</code>	Cold: Chip and CPU restart, e.g. power cycling
<code>VTSS_RESTART_COOL</code>	Cool: Chip and CPU restart done by CPU
<code>VTSS_RESTART_WARM</code>	Warm: CPU restart only

Definition at line 601 of file vtss\_init\_api.h.

### 6.15.5 Function Documentation

#### 6.15.5.1 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.

**6.15.5.2 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.

**6.15.5.3 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.

#### 6.15.5.4 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.

#### 6.15.5.5 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.

#### 6.15.5.6 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.

**6.15.5.7 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.

**6.15.5.8 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.

**6.15.5.9 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.

## 6.16 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](#)

- 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\_UPSID - 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_TYPE_ETYPE`, `VTSS_VCE_TYPE_LLC`, `VTSS_VCE_TYPE_SNAP`,  
`VTSS_VCE_TYPE_IPV4`, `VTSS_VCE_TYPE_IPV6` }
- VCE frame type.
- enum `vtss_mirror_tag_t`{ `VTSS_MIRROR_TAG_NONE`, `VTSS_MIRROR_TAG_C`, `VTSS_MIRROR_TAG_S`,  
`VTSS_MIRROR_TAG_S_CUSTOM` }
- Mirror port configuration.
- enum `vtss_eps_port_type_t`{ `VTSS_EPS_PORT_1_PLUS_1`, `VTSS_EPS_PORT_1_FOR_1` }
- Port protection type.
- enum `vtss_eps_selector_t`{ `VTSS_EPS_SELECTOR_WORKING`, `VTSS_EPS_SELECTOR_PROTECTION` }
- EPS selector.
- enum `vtss_erps_state_t`{ `VTSS_ERPS_STATE_FORWARDING`, `VTSS_ERPS_STATE_DISCARDING` }
- ERPS state.
- enum `vtss_vstax_topology_type_t`{ `VTSS_VSTAX_TOPOLOGY_CHAIN`, `VTSS_VSTAX_TOPOLOGY_RING` }
- VStaX topology type.

## Functions

- `vtss_rc vtss_mac_table_add` (const `vtss_inst_t` inst, const `vtss_mac_table_entry_t` \*const entry)  
*Add MAC address entry.*
- `vtss_rc vtss_mac_table_del` (const `vtss_inst_t` inst, const `vtss_vid_mac_t` \*const vid\_mac)  
*Delete MAC address entry.*
- `vtss_rc vtss_mac_table_get` (const `vtss_inst_t` inst, const `vtss_vid_mac_t` \*const vid\_mac, `vtss_mac_table_entry_t` \*const entry)  
*Get MAC address entry.*
- `vtss_rc vtss_mac_table_get_next` (const `vtss_inst_t` inst, const `vtss_vid_mac_t` \*const vid\_mac, `vtss_mac_table_entry_t` \*const entry)  
*Lookup next MAC address entry.*
- `vtss_rc vtss_mac_table_age_time_get` (const `vtss_inst_t` inst, `vtss_mac_table_age_time_t` \*const age\_time)  
*Get MAC address table age time.*
- `vtss_rc vtss_mac_table_age_time_set` (const `vtss_inst_t` inst, const `vtss_mac_table_age_time_t` age\_time)  
*Set MAC address table age time.*
- `vtss_rc vtss_mac_table_age` (const `vtss_inst_t` inst)  
*Do age scan of the MAC address table. This should be done periodically with interval T/2, where T is the age timer.*
- `vtss_rc vtss_mac_table_vlan_age` (const `vtss_inst_t` inst, const `vtss_vid_t` vid)  
*Do VLAN specific age scan of the MAC address table.*
- `vtss_rc vtss_mac_table_flush` (const `vtss_inst_t` inst)  
*Flush MAC address table, i.e. remove all unlocked entries.*
- `vtss_rc vtss_mac_table_port_flush` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no)  
*Delete MAC address entries learned on port.*
- `vtss_rc vtss_mac_table_vlan_flush` (const `vtss_inst_t` inst, const `vtss_vid_t` vid)  
*Delete MAC address entries learned on VLAN ID.*
- `vtss_rc vtss_mac_table_vlan_port_flush` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_vid_t` vid)  
*Delete MAC address entries learned on port and VLAN ID.*
- `vtss_rc vtss_mac_table_upsid_flush` (const `vtss_inst_t` inst, const `vtss_vstax_upsid_t` upsid)

- Delete MAC address entries learned on UPSID.*

  - `vtss_rc vtss_mac_table_upsid_upspn_flush` (const `vtss_inst_t` inst, const `vtss_vstax_upsid_t` upsid, const `vtss_vstax_upspn_t` upspn)
- Delete MAC address entries learned on (UPSID, UPSPN).*

  - `vtss_rc vtss_mac_table_glag_add` (const `vtss_inst_t` inst, const `vtss_mac_table_entry_t` \*const entry, const `vtss_glag_no_t` glag\_no)
- Learn MAC address entry on GLAG.*

  - `vtss_rc vtss_mac_table_glag_flush` (const `vtss_inst_t` inst, const `vtss_glag_no_t` glag\_no)
- Delete MAC address entries learned on GLAG.*

  - `vtss_rc vtss_mac_table_vlan_glag_flush` (const `vtss_inst_t` inst, const `vtss_glag_no_t` glag\_no, const `vtss_vid_t` vid)
- Delete MAC address entries learned on GLAG and VID.*

  - `vtss_rc vtss_mac_table_status_get` (const `vtss_inst_t` inst, `vtss_mac_table_status_t` \*const status)
- Get MAC address table status.*

  - `vtss_rc vtss_learn_port_mode_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_learn_mode_t` \*const mode)
- Get the learn mode for a port.*

  - `vtss_rc vtss_learn_port_mode_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_learn_mode_t` \*const mode)
- Set the learn mode for a port.*

  - `vtss_rc vtss_port_state_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `BOOL` \*const state)
- Get port operational state.*

  - `vtss_rc vtss_port_state_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `BOOL` state)
- Set port operational state.*

  - `vtss_rc vtss_stp_port_state_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_stp_state_t` \*const state)
- Get Spanning Tree state for a port.*

  - `vtss_rc vtss_stp_port_state_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_stp_state_t` state)
- Set Spanning Tree state for a port.*

  - `vtss_rc vtss_mstp_vlan_msti_get` (const `vtss_inst_t` inst, const `vtss_vid_t` vid, `vtss_msti_t` \*const msti)
- Get MSTP instance mapping for a VLAN.*

  - `vtss_rc vtss_mstp_vlan_msti_set` (const `vtss_inst_t` inst, const `vtss_vid_t` vid, const `vtss_msti_t` msti)
- Set MSTP instance mapping for a VLAN.*

  - `vtss_rc vtss_mstp_port_msti_state_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_msti_t` msti, `vtss_stp_state_t` \*const state)
- Get MSTP state for a port and MSTP instance.*

  - `vtss_rc vtss_mstp_port_msti_state_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_msti_t` msti, const `vtss_stp_state_t` state)
- Set MSTP state for a port and MSTP instance.*

  - `vtss_rc vtss_vlan_conf_get` (const `vtss_inst_t` inst, `vtss_vlan_conf_t` \*const conf)
- Get VLAN configuration.*

  - `vtss_rc vtss_vlan_conf_set` (const `vtss_inst_t` inst, const `vtss_vlan_conf_t` \*const conf)
- Set VLAN configuration.*

  - `vtss_rc vtss_vlan_port_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_vlan_port_conf_t` \*const conf)
- Get VLAN mode for port.*

  - `vtss_rc vtss_vlan_port_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_vlan_port_conf_t` \*const conf)
- Set VLAN mode for port.*

  - `vtss_rc vtss_vlan_port_members_get` (const `vtss_inst_t` inst, const `vtss_vid_t` vid, `BOOL` member[`VTSS_PORT_ARRAY_SIZE`])
- Get VLAN membership.*

- `vtss_rc vtss_vlan_port_members_set` (const `vtss_inst_t` inst, const `vtss_vid_t` vid, const `BOOL` member[`VTSS_PORT_ARRAY_SIZE`])  
*Set VLAN membership.*
- `vtss_rc vtss_vlan_vid_conf_get` (const `vtss_inst_t` inst, const `vtss_vid_t` vid, `vtss_vlan_vid_conf_t` \*const conf)  
*Get VLAN ID configuration.*
- `vtss_rc vtss_vlan_vid_conf_set` (const `vtss_inst_t` inst, const `vtss_vid_t` vid, const `vtss_vlan_vid_conf_t` \*const conf)  
*Set VLAN ID configuration.*
- `vtss_rc vtss_vlan_tx_tag_get` (const `vtss_inst_t` inst, const `vtss_vid_t` vid, `vtss_vlan_tx_tag_t` tx←tag[`VTSS_PORT_ARRAY_SIZE`])  
*Get VLAN Tx tagging configuration.*
- `vtss_rc vtss_vlan_tx_tag_set` (const `vtss_inst_t` inst, const `vtss_vid_t` vid, const `vtss_vlan_tx_tag_t` tx←tag[`VTSS_PORT_ARRAY_SIZE`])  
*Get VLAN Tx tagging configuration.*
- `vtss_rc vtss_vcl_port_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_vcl_port_conf_t` \*const conf)  
*Get VCL port configuration.*
- `vtss_rc vtss_vcl_port_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_vcl_port_conf_t` \*const conf)  
*Get VCL port configuration.*
- `vtss_rc vtss_vce_init` (const `vtss_inst_t` inst, const `vtss_vce_type_t` type, `vtss_vce_t` \*const vce)  
*Initialize VCE to default values.*
- `vtss_rc vtss_vce_add` (const `vtss_inst_t` inst, const `vtss_vce_id_t` vce\_id, const `vtss_vce_t` \*const vce)  
*Add/modify VCE.*
- `vtss_rc vtss_vce_del` (const `vtss_inst_t` inst, const `vtss_vce_id_t` vce\_id)  
*Delete VCE.*
- `vtss_rc vtss_vlan_trans_group_add` (const `vtss_inst_t` inst, const `u16` group\_id, const `vtss_vid_t` vid, const `vtss_vid_t` trans\_vid)  
*Create VLAN Translation Group entry.*
- `vtss_rc vtss_vlan_trans_group_del` (const `vtss_inst_t` inst, const `u16` group\_id, const `vtss_vid_t` vid)  
*Delete VLAN Translation Group entry.*
- `vtss_rc vtss_vlan_trans_group_get` (const `vtss_inst_t` inst, `vtss_vlan_trans_grp2vlan_conf_t` \*conf, `BOOL` next)  
*Get VLAN Translation Group entry.*
- `vtss_rc vtss_vlan_trans_group_to_port_set` (const `vtss_inst_t` inst, const `vtss_vlan_trans_port2grp_conf_t` \*conf)  
*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`])

- `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: $\cdot\cdot\cdot$ :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.*

### 6.16.1 Detailed Description

Layer 2 API.

This header file describes Layer 2 switching functions

### 6.16.2 Macro Definition Documentation

#### 6.16.2.1 VTSS\_MAC\_ADDRS

```
#define VTSS_MAC_ADDRS 32768
```

Number of MAC addresses

Definition at line 93 of file vtss\_l2\_api.h.

### 6.16.2.2 VTSS\_VSTAX\_UPSIDS

```
#define VTSS_VSTAX_UPSIDS (32)
```

Number of UPSIDs

Definition at line 102 of file vtss\_l2\_api.h.

### 6.16.2.3 VTSS\_VSTAX\_UPSID\_START

```
#define VTSS_VSTAX_UPSID_START ( 0 )
```

First UPSID value

Definition at line 103 of file vtss\_l2\_api.h.

### 6.16.2.4 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.

### 6.16.2.5 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.

### 6.16.2.6 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.

### 6.16.2.7 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.

### 6.16.2.8 VTSS\_UPSPN\_CPU

```
#define VTSS_UPSPN_CPU 0xffffffff
```

MAC address entry is from CPU

Definition at line 114 of file vtss\_l2\_api.h.

### 6.16.2.9 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.

### 6.16.2.10 VTSS\_MSTIS

```
#define VTSS_MSTIS (65)
```

Number of MSTP instances

Definition at line 514 of file vtss\_l2\_api.h.

### 6.16.2.11 VTSS\_MSTI\_START

```
#define VTSS_MSTI_START (0)
```

MSTI start number

Definition at line 515 of file vtss\_l2\_api.h.

### 6.16.2.12 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.

### 6.16.2.13 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.

### 6.16.2.14 VTSS\_VCL\_IDS

```
#define VTSS_VCL_IDS 256
```

Number of VCLs

Definition at line 922 of file vtss\_l2\_api.h.

### 6.16.2.15 VTSS\_VCL\_ID\_START

```
#define VTSS_VCL_ID_START 0
```

VCL ID start number

Definition at line 923 of file vtss\_l2\_api.h.

### 6.16.2.16 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.

**6.16.2.17 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.

**6.16.2.18 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.

**6.16.2.19 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.

**6.16.2.20 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.

**6.16.2.21 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.

### 6.16.2.22 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.

### 6.16.2.23 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.

### 6.16.2.24 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.

### 6.16.2.25 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.

### 6.16.2.26 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.

### 6.16.2.27 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.

### 6.16.2.28 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.

### 6.16.2.29 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.

### 6.16.2.30 VTSS\_PVLANS

```
#define VTSS_PVLANS (VTSS_PORTS)
```

Number of PVLANS

Definition at line 1304 of file vtss\_l2\_api.h.

### 6.16.2.31 VTSS\_PVLAN\_NO\_START

```
#define VTSS_PVLAN_NO_START (0)
```

PVLAN start number

Definition at line 1305 of file vtss\_l2\_api.h.

### 6.16.2.32 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.

### 6.16.2.33 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.

### 6.16.2.34 VTSS\_PVLAN\_NO\_DEFAULT

```
#define VTSS_PVLAN_NO_DEFAULT (0)
```

Default PVLAN

Definition at line 1308 of file vtss\_l2\_api.h.

### 6.16.2.35 VTSS\_ERPIS

```
#define VTSS_ERPIS (64)
```

Number of ERPS instances

Definition at line 2228 of file vtss\_l2\_api.h.

### 6.16.2.36 VTSS\_ERPI\_START

```
#define VTSS_ERPI_START (0)
```

ERPI start number

Definition at line 2229 of file vtss\_l2\_api.h.

### 6.16.2.37 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.

### 6.16.2.38 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.

## 6.16.3 Typedef Documentation

### 6.16.3.1 vtss\_vt\_id\_t

```
typedef u64 vtss_vt_id_t
```

VLAN Translation ID

Definition at line 1095 of file vtss\_l2\_api.h.

## 6.16.4 Enumeration Type Documentation

### 6.16.4.1 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.

**6.16.4.2 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.

**6.16.4.3 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.

**6.16.4.4 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.

#### 6.16.4.5 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.

#### 6.16.4.6 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.

#### 6.16.4.7 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.

#### 6.16.4.8 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.

#### 6.16.4.9 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.

### 6.16.5 Function Documentation

#### 6.16.5.1 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.

**6.16.5.2 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.

**6.16.5.3 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.

#### 6.16.5.4 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.

#### 6.16.5.5 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.

#### 6.16.5.6 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.

**6.16.5.7 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.

**6.16.5.8 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.

### 6.16.5.9 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.

### 6.16.5.10 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.

### 6.16.5.11 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.

**6.16.5.12 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.

**6.16.5.13 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.

**6.16.5.14 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.

### 6.16.5.15 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.

### 6.16.5.16 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.

**6.16.5.17 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.

**6.16.5.18 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.

**6.16.5.19 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.

### 6.16.5.20 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.

### 6.16.5.21 vtss\_port\_state\_get()

```
vtss_rc vtss_port_state_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    BOOL *const state )
```

Get port operational state.

#### Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>state</i>	[OUT] Port state, TRUE if link is up.

Generated by Doxygen

**Returns**

Return code.

**6.16.5.22 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.

**6.16.5.23 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.

## 6.16.5.24 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.

## 6.16.5.25 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.

## 6.16.5.26 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.

**6.16.5.27 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.

**6.16.5.28 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.

**6.16.5.29 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.

**6.16.5.30 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.

**6.16.5.31 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.

**6.16.5.32 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.

**6.16.5.33 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.

**6.16.5.34 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.

**6.16.5.35 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.

### 6.16.5.36 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.

### 6.16.5.37 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.

### 6.16.5.38 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.

**6.16.5.39 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.

**6.16.5.40 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.

**6.16.5.41 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.

**6.16.5.42 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.

#### 6.16.5.43 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.

#### 6.16.5.44 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.

#### 6.16.5.45 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.

**6.16.5.46 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.

**6.16.5.47 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.

**6.16.5.48 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.

**6.16.5.49 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.

## 6.16.5.50 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.

## 6.16.5.51 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.

## 6.16.5.52 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.

**6.16.5.53 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.

**6.16.5.54 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.

**6.16.5.55 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.

**6.16.5.56 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.

### 6.16.5.57 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.

### 6.16.5.58 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.

### 6.16.5.59 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.

**6.16.5.60 vtss\_sfflow\_port\_conf\_set()**

```
vtss_rc vtss_sfflow_port_conf_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_sfflow_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.

**6.16.5.61 vtss\_sfflow\_sampling\_rate\_convert()**

```
vtss_rc vtss_sfflow_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.

**6.16.5.62 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.

**6.16.5.63 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.

**6.16.5.64 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.

**6.16.5.65 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.

**6.16.5.66 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.

**6.16.5.67 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.

**6.16.5.68 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.

**6.16.5.69 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.

**6.16.5.70 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.

**6.16.5.71 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.

**6.16.5.72 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.

**6.16.5.73 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.

#### 6.16.5.74 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.

#### 6.16.5.75 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.

#### 6.16.5.76 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.

**6.16.5.77 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.

**6.16.5.78 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.

**6.16.5.79 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.

### 6.16.5.80 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.

### 6.16.5.81 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.

**6.16.5.82 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.

**6.16.5.83 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.

**6.16.5.84 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.

**6.16.5.85 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.

**6.16.5.86 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.

## 6.16.5.87 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.

## 6.16.5.88 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.

## 6.16.5.89 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.

**6.16.5.90 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.

**6.16.5.91 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.

## 6.16.5.92 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.

## 6.16.5.93 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.

## 6.16.5.94 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.

**6.16.5.95 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.

**6.16.5.96 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.

**6.16.5.97 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.

**6.16.5.98 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.

### 6.16.5.99 `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.

### 6.16.5.100 `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.

### 6.16.5.101 `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.

**6.16.5.102 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.

**6.16.5.103 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.

**6.16.5.104 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.

**6.16.5.105 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.

**6.17 vtss\_api/include/vtss\_l3\_api.h File Reference**

L3 routing API.

```
#include <vtss/api/types.h>
```

### 6.17.1 Detailed Description

L3 routing API.

This header file describes L3 IPv4/IPv6 hardware assisted routing functions.

## 6.18 vtss\_api/include/vtss\_mac10g\_api.h File Reference

MAC10G API.

```
#include <vtss/api/types.h>
```

### 6.18.1 Detailed Description

MAC10G API.

## 6.19 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](#)  
*GPIO configuration for a group.*
- struct [vtss\\_sgpio\\_port\\_data\\_t](#)

- *SGPIO read data for a port.*
- struct `vtss_intr_t`  
*Interrupt source structure.*
- 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` 64  
*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` 3  
*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`
    - GPIO configured mode.*
    - 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 }
  - enum `vtss_sgpio_bmode_t`
    - GPIO output mode.*
    - VTSS\_SGPIO\_BMODE\_TOGGLE, VTSS\_SGPIO\_BMODE\_0\_625, VTSS\_SGPIO\_BMODE\_1\_25, VTSS\_SGPIO\_BMODE\_2\_5, VTSS\_SGPIO\_BMODE\_5 }
  - enum `vtss_irq_t`
    - GPIO blink mode.*
    - 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 }
  - enum `vtss_fan_pwd_freq_t`
    - Interrupt sources.*
    - 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 }
  - 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\_intr\_mask\_set** (const `vtss_inst_t` inst, `vtss_intr_t` \*mask)
  - Set the interrupt mask.*
- **vtss\_rc vtss\_intr\_status\_get** (const `vtss_inst_t` inst, `vtss_intr_t` \*status)
  - Get the interrupt status for all enabled sources. The interrupt status bit is cleared by the function.*
- **vtss\_rc vtss\_intr\_pol\_negation** (const `vtss_inst_t` inst)
  - This will negate polarity on fast link fail detection signals when active. This is only intended to be used on Luton26 RevB as a work around for the Atom PHY interrupt always active problem.*
- **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.*

### **6.19.1 Detailed Description**

## Miscellaneous API.

This header file describes miscellaneous API functions

## 6.19.2 Macro Definition Documentation

#### 6.19.2.1 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.

### **6.19.2.2 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.

### 6.19.3 Typedef Documentation

#### **6.19.3.1 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.

## 6.19.4 Enumeration Type Documentation

### 6.19.4.1 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.

### 6.19.4.2 vtss\_trace\_group\_t

```
enum vtss_trace_group_t
```

Trace groups.

#### Enumerator

VTSS_TRACE_GROUP_DEFAULT	Default trace group
VTSS_TRACE_GROUP_PORT	Port control
VTSS_TRACE_GROUP_PHY	PHY control
VTSS_TRACE_GROUP_PACKET	Packet control
VTSS_TRACE_GROUP_AFI	AFI
VTSS_TRACE_GROUP_QOS	Quality of Service
VTSS_TRACE_GROUP_L2	Layer 2
VTSS_TRACE_GROUP_L3	Layer 3
VTSS_TRACE_GROUP_SECURITY	Security
VTSS_TRACE_GROUP_EVC	Ethernet Virtual Connections
VTSS_TRACE_GROUP_FDMA_NORMAL	Frame DMA Extraction and Injection when interrupts/scheduler is enabled
VTSS_TRACE_GROUP_FDMA_IRQ	Frame DMA when interrupts/scheduler is disabled Daytona layers are placed before the PHY layer, otherwise they are not shown in the CLI commands
VTSS_TRACE_GROUP_REG_CHECK	Register access errors (must be able to print when interrupts/scheduler is disabled)
VTSS_TRACE_GROUP MPLS	MPLS
VTSS_TRACE_GROUP_HQOS	Hierarchical Quality of Service
VTSS_TRACE_GROUP_MACSEC	MACSEC control
VTSS_TRACE_GROUP_VCAP	VCAP
VTSS_TRACE_GROUP_OAM	OAM
VTSS_TRACE_GROUP_TS	Timestamping
VTSS_TRACE_GROUP_COUNT	Number of trace groups

Definition at line 52 of file vtss\_misc\_api.h.

#### 6.19.4.3 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.

#### 6.19.4.4 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.

#### 6.19.4.5 vtss\_debug\_group\_t

enum `vtss_debug_group_t`

Debug function group.

Enumerator

VTSS_DEBUG_GROUP_ALL	All groups
----------------------	------------

## Enumerator

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
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.

#### 6.19.4.6 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
VTSS_GPIO_ALT_2	Alternate function 2

Definition at line 567 of file vtss\_misc\_api.h.

## 6.19.4.7 vtss\_sgpiemode\_t

```
enum vtss_sgpiemode_t
```

SGPIO output mode.

## Enumerator

VTSS_SGPIO_MODE_OFF	Off
VTSS_SGPIO_MODE_ON	On
VTSS_SGPIO_MODE_0	Mode 0
VTSS_SGPIO_MODE_1	Mode 1
VTSS_SGPIO_MODE_0_ACTIVITY	Mode 0 when link activity
VTSS_SGPIO_MODE_1_ACTIVITY	Mode 1 when link activity
VTSS_SGPIO_MODE_0_ACTIVITY_INV	Mode 0 when link activity, inversed polarity
VTSS_SGPIO_MODE_1_ACTIVITY_INV	Mode 1 when link activity, inversed polarity

Definition at line 766 of file vtss\_misc\_api.h.

## 6.19.4.8 vtss\_sgpiobmode\_t

```
enum vtss_sgpiobmode_t
```

SGPIO blink mode.

## Enumerator

VTSS_SGPIO_BMODE_TOGGLE	Burst toggle (mode 1 only)
VTSS_SGPIO_BMODE_0_625	0.625 Hz (mode 0 only)
VTSS_SGPIO_BMODE_1_25	1.25 Hz
VTSS_SGPIO_BMODE_2_5	2.5 Hz
VTSS_SGPIO_BMODE_5	5 Hz

Definition at line 779 of file vtss\_misc\_api.h.

#### 6.19.4.9 vtss\_irq\_t

enum `vtss_irq_t`

Interrupt sources.

Enumerator

<code>VTSS_IRQ_XTR</code>	Frame Extraction Ready(register-based)
<code>VTSS_IRQ_FDMA_XTR</code>	Frame Extraction Ready (FDMA-based)
<code>VTSS_IRQ_SOFTWARE</code>	Software IRQ
<code>VTSS_IRQ_PTP_RDY</code>	PTP Timestamp Ready
<code>VTSS_IRQ_PTP_SYNC</code>	PTP Synchronization IRQ
<code>VTSS_IRQ_EXT1</code>	EXT1 IRQ
<code>VTSS_IRQ_OAM</code>	OAM IRQ
<code>VTSS_IRQ_MAX</code>	Maximum IRQ Source

Definition at line 957 of file vtss\_misc\_api.h.

#### 6.19.4.10 vtss\_eee\_state\_select\_t

enum `vtss_eee_state_select_t`

EEE port state change what? (JR only)

Enumerator

<code>VTSS_EEE_STATE_SELECT_LPI</code>	Change LPI signal.
<code>VTSS_EEE_STATE_SELECT_SCH</code>	Change scheduler enable.
<code>VTSS_EEE_STATE_SELECT_FP</code>	Change frame mirroring flag.
<code>VTSS_EEE_STATE_SELECT_INTR_ENA</code>	Enable analyzer interrupts.
<code>VTSS_EEE_STATE_SELECT_INTR_ACK</code>	Acknowledge analyzer interrupts.

Definition at line 1230 of file vtss\_misc\_api.h.

### 6.19.5 Function Documentation

#### 6.19.5.1 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

<i>group</i>	[IN] Trace group
<i>conf</i>	[OUT] Trace group configuration.

##### Returns

Return code.

#### 6.19.5.2 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.

#### 6.19.5.3 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.

**6.19.5.4 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.

#### 6.19.5.5 vtss\_debug\_info\_get()

```
vtss_rc vtss_debug_info_get (
    const vtss_debug_info_t *const info )
```

Get default debug information structure.

##### Parameters

<i>info</i>	[OUT] Debug information
-------------	-------------------------

##### Returns

Return code.

#### 6.19.5.6 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.

#### 6.19.5.7 vtss\_callout\_lock()

```
void vtss_callout_lock (
    const vtss_api_lock_t *const lock )
```

Lock API access.

##### Parameters

<i>lock</i>	[IN] Lock information
-------------	-----------------------

### 6.19.5.8 vtss\_callout\_unlock()

```
void vtss_callout_unlock (
    const vtss_api_lock_t *const lock )
```

Unlock API access.

#### Parameters

<i>lock</i>	[IN] Lock information
-------------	-----------------------

### 6.19.5.9 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.

### 6.19.5.10 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.

**6.19.5.11 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.

**6.19.5.12 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.

### 6.19.5.13 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.

### 6.19.5.14 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.

### 6.19.5.15 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.

**6.19.5.16 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.

**6.19.5.17 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.

**6.19.5.18 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.

**6.19.5.19 vtss\_dev\_all\_event\_poll()**

```
vtss_rc vtss_dev_all_event_poll (
    const vtss_inst_t inst,
    const vtss_dev_all_event_poll_t poll_type,
    vtss_dev_all_event_type_t *const ev_mask )
```

DEV\_ALL polling function called at by interrupt or periodically.

**Parameters**

<i>inst</i>	[IN] Target instance reference.
<i>poll_type</i>	[IN] Polling type
<i>ev_mask</i>	[OUT] Event type mask array of active events for all ports - must be of size VTSS_PORT_ARRAY_SIZE

**Note**

The *ev\_mask* parameter can be either a single event\_type or multiple event types (or *VTSS\_EVTYPE\_ALL*). If invoked by a processor interrupt signal, the type of event to check for may be narrowed in to specific events.

**Returns**

Return code.

**6.19.5.20 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.

**6.19.5.21 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.

### 6.19.5.22 `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.

### 6.19.5.23 `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.

### 6.19.5.24 `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.

### 6.19.5.25 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.

### 6.19.5.26 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.

**6.19.5.27 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.

**6.19.5.28 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.

**6.19.5.29 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 SGPIO data.

**Parameters**

<i>inst</i>	[IN] Target instance reference.
<i>chip_no</i>	[IN] Chip number (if multi-chip instance).
<i>group</i>	[IN] SGPIO group.
<i>data</i>	[OUT] SGPIO data.

**Returns**

Return code.

**6.19.5.30 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 SGPIO 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] SGPIO group.
<i>bit</i>	[IN] SGPIO 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.

**6.19.5.31 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.

**6.19.5.32 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.

**6.19.5.33 vtss\_intr\_mask\_set()**

```
vtss_rc vtss_intr_mask_set (
    const vtss_inst_t inst,
    vtss_intr_t * mask )
```

Set the interrupt mask.

**Parameters**

<i>inst</i>	[IN] Target instance reference.
<i>mask</i>	[IN] Pointer to mask structure.

**Returns**

Return code.

**6.19.5.34 vtss\_intr\_status\_get()**

```
vtss_rc vtss_intr_status_get (
    const vtss_inst_t inst,
    vtss_intr_t * status )
```

Get the interrupt status for all enabled sources. The interrupt status bit is cleared by the function.

**Parameters**

<i>inst</i>	[IN] Target instance reference.
<i>status</i>	[OUT] Pointer to a structure with status of all enabled interrupt sources.

**Returns**

Return code.

**6.19.5.35 vtss\_intr\_pol\_negation()**

```
vtss_rc vtss_intr_pol_negation (
    const vtss_inst_t inst )
```

This vil negate polarity on fast link fail detection signals when active. This is only intended to be used on Luton26 RevB as a work around for the Atom PHY interrupt always active problem.

**Parameters**

<i>inst</i>	[IN] Target instance reference.
-------------	---------------------------------

**Returns**

Return code.

**6.19.5.36 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.

**6.19.5.37 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.

**6.19.5.38 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.

**6.19.5.39 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.

**6.19.5.40 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

#### 6.19.5.41 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

<i>cb</i>	pointer to callback function
-----------	------------------------------

#### 6.19.5.42 vtss\_temp\_sensor\_init()

```
vtss_rc vtss_temp_sensor_init (
    const vtss_inst_t inst,
    const BOOL enable )
```

Initialize the temperature sensor.

#### Parameters

<i>inst</i>	[IN] Target instance reference
<i>enable</i>	[IN] Set to true if sensor shall be active else false

#### Returns

Return code.

#### 6.19.5.43 vtss\_temp\_sensor\_get()

```
vtss_rc vtss_temp_sensor_get (
    const vtss_inst_t inst,
    i16 * temperature )
```

Read temperature sensor value.

**Parameters**

<i>inst</i>	[IN] Target instance reference
<i>temperature</i>	[OUT] Temperature from sensor (range from -46 to 135 degC)

**Returns**

Return code.

**6.19.5.44 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.

**6.19.5.45 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.

#### 6.19.5.46 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.

#### 6.19.5.47 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.

#### 6.19.5.48 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.

**6.19.5.49 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.

**6.19.5.50 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. Generated by Doxygen

**Returns**

Return code.

**6.19.5.51 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.

**6.20 vtss\_api/include/vtss\_mpls\_api.h File Reference**

MPLS API.

```
#include <vtss/api/types.h>
```

**6.20.1 Detailed Description**

MPLS API.

This header file describes the MPLS functions

## 6.21 vtss\_api/include/vtss\_oam\_api.h File Reference

OAM API.

```
#include <vtss/api/types.h>
```

### 6.21.1 Detailed Description

OAM API.

This header file describes Y.1731/IEEE802.1ag OAM functions.

## 6.22 vtss\_api/include/vtss\_oha\_api.h File Reference

OHA API.

```
#include <vtss/api/types.h>
```

### 6.22.1 Detailed Description

OHA API.

## 6.23 vtss\_api/include/vtss\_os.h File Reference

OS Layer API.

```
#include <vtss_os_ecos.h>
```

### 6.23.1 Detailed Description

OS Layer API.

This header file includes the OS specific header file

## 6.24 vtss\_api/include/vtss\_os\_custom.h File Reference

OS custom header file.

```
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
```

## Macros

- #define `uint` unsigned int
- #define `ulong` unsigned long
- #define `VTSS_MSLEEP`(msec) <your function>
- #define `VTSS_MTIMER_START`(pTimer, msec) <your impl>
- #define `VTSS_MTIMER_TIMEOUT`(pTimer) <your impl>
- #define `VTSS_MTIMER_CANCEL`(pTimer) <your impl>
- #define `VTSS_DIV64`(dividend, divisor) <your impl>
- #define `VTSS_MOD64`(dividend, divisor) <your impl>
- #define `VTSS_LABS`(arg) <your impl>
- #define `VTSS_LLABS`(arg) <your impl>
- #define `VTSS_OS_CTZ`(val32) <your impl>
- #define `VTSS_OS_CTZ64`(val64) <your impl>
- #define `VTSS_OS_MALLOC`(size, flags) <your impl>
- #define `VTSS_OS_FREE`(ptr, flags) <your impl>
- #define `VTSS_OS_RAND`() <your impl>

## Typedefs

- typedef int `vtss_mtimer_t`

### 6.24.1 Detailed Description

OS custom header file.

This file is a skeleton to be replaced by for customer specific OS.

### 6.24.2 Macro Definition Documentation

#### 6.24.2.1 `uint`

```
#define uint unsigned int
```

Define API uint type - unsigned 16 bits

Definition at line 36 of file vtss\_os\_custom.h.

#### 6.24.2.2 `ulong`

```
#define ulong unsigned long
```

Define API ulong - unsigned 32 bits

Definition at line 37 of file vtss\_os\_custom.h.

### 6.24.2.3 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.

### 6.24.2.4 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.

### 6.24.2.5 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.

### 6.24.2.6 VTSS\_MTIMER\_CANCEL

```
#define VTSS_MTIMER_CANCEL(  
    pTimer ) <your impl>
```

Stop the timer

Definition at line 46 of file vtss\_os\_custom.h.

### 6.24.2.7 VTSS\_DIV64

```
#define VTSS_DIV64(  
    dividend,  
    divisor ) <your impl>
```

support for 64 bit division

Definition at line 48 of file vtss\_os\_custom.h.

#### 6.24.2.8 VTSS\_MOD64

```
#define VTSS_MOD64 (
    dividend,
    divisor ) <your impl>
```

support for 64 bit division

Definition at line 49 of file vtss\_os\_custom.h.

#### 6.24.2.9 VTSS\_LABS

```
#define VTSS_LABS (
    arg ) <your impl>
```

long to abs

Definition at line 50 of file vtss\_os\_custom.h.

#### 6.24.2.10 VTSS\_LLABS

```
#define VTSS_LLABS (
    arg ) <your impl>
```

long long to abs

Definition at line 51 of file vtss\_os\_custom.h.

#### 6.24.2.11 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.

### 6.24.2.12 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.

### 6.24.2.13 VTSS\_OS\_MALLOC

```
#define VTSS_OS_MALLOC( size, flags ) <your impl>
```

Request OS to allocate size bytes of memory.

The first argument is the number of bytes that must be allocated. Type is `size_t`.

The second argument is a mask of flags that the implementation must obey. Type is `vtss_mem_flags_t`.

The returned pointer should be at least 8-byte aligned, to make it suitable for a struct.

Definition at line 87 of file vtss\_os\_custom.h.

### 6.24.2.14 VTSS\_OS\_FREE

```
#define VTSS_OS_FREE( ptr, flags ) <your impl>
```

Request OS to free memory previously allocated with `VTSS_OS_MALLOC()`.

The first argument is the pointer previously obtained with a call to `VTSS_OS_MALLOC()`. Type is `void *`.

The second argument is a mask of flags identical to those passed to `VTSS_OS_MALLOC()` when the memory was requested.

Definition at line 101 of file vtss\_os\_custom.h.

### 6.24.2.15 VTSS\_OS\_RAND

```
#define VTSS_OS_RAND( ) <your impl>
```

Wrap of call to `rand()` defined in `stdlib.h`

Definition at line 106 of file vtss\_os\_custom.h.

### 6.24.3 Typedef Documentation

#### 6.24.3.1 vtss\_mtimer\_t

```
typedef int vtss_mtimer_t
```

Timer

Definition at line 43 of file vtss\_os\_custom.h.

## 6.25 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\_NTOHL(v1) (((v1) >> 24) & 0x000000FF) | (((v1) >> 8) & 0x0000FF00) | (((v1) << 8) & 0xFF000000) | (((v1) << 24) & 0x000000FF)
- VTSS\_OS\_BIG\_ENDIAN: If undefined, we're running little endian. If defined we're running big endian.
- #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.*

### 6.25.1 Detailed Description

#### eCos OS API

This header file describes OS functions for eCos

### 6.25.2 Macro Definition Documentation

#### 6.25.2.1 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.

### 6.25.2.2 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.

### 6.25.2.3 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.

### 6.25.2.4 VTSS\_MTIMER\_TIMEOUT

```
#define VTSS_MTIMER_TIMEOUT( pTimer ) (cyg_current_time() > *(pTimer))
```

Timer timeout

Definition at line 55 of file vtss\_os\_ecos.h.

### 6.25.2.5 VTSS\_MTIMER\_CANCEL

```
#define VTSS_MTIMER_CANCEL( pTimer )
```

No action in this implementation.

Definition at line 56 of file vtss\_os\_ecos.h.

### 6.25.2.6 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.

### 6.25.2.7 VTSS\_DIV64

```
#define VTSS_DIV64(  
    dividend,  
    divisor ) ((dividend) / (divisor))
```

support for 64 bit division

Definition at line 73 of file vtss\_os\_ecos.h.

### 6.25.2.8 VTSS\_MOD64

```
#define VTSS_MOD64(  
    dividend,  
    divisor ) ((dividend) % (divisor))
```

support for 64 bit division

Definition at line 74 of file vtss\_os\_ecos.h.

### 6.25.2.9 VTSS\_LABS

```
#define VTSS_LABS(  
    arg ) labs(arg)
```

long to abs

Definition at line 75 of file vtss\_os\_ecos.h.

### 6.25.2.10 VTSS\_LLabs

```
#define VTSS_LLabs(  
    arg ) llabs(arg)
```

long long to abs

Definition at line 76 of file vtss\_os\_ecos.h.

### 6.25.2.11 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`.

### 6.25.2.12 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`.

### 6.25.2.13 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`.

### 6.25.2.14 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.

### 6.25.2.15 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.

### 6.25.2.16 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.

### 6.25.2.17 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.

### 6.25.2.18 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.

### 6.25.2.19 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.

### 6.25.2.20 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.

### 6.25.2.21 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.

### 6.25.2.22 VTSS\_OS\_NTOHL

```
#define VTSS_OS_NTOHL(
    v1 ) (((v1) >> 24) & 0x000000FF) | (((v1) >> 8) & 0x0000FF00) | (((v1) << 8) &
0x00FF0000) | (((v1) << 24) & 0xFF000000))
```

VTSS\_OS\_BIG\_ENDIAN: If undefined, we're running little endian. If defined we're running big endian.

Convert a 32-bit value from network to host order

Definition at line 226 of file vtss\_os\_ecos.h.

### 6.25.2.23 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.

### 6.25.2.24 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.

### 6.25.2.25 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.

### 6.25.2.26 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.

### 6.25.2.27 VTSS\_OS\_INTERRUPT\_DISABLE

```
#define VTSS_OS_INTERRUPT_DISABLE(
    flags ) NOT_NEEDED
```

Disable interrupts.

Definition at line 276 of file vtss\_os\_ecos.h.

### 6.25.2.28 VTSS\_OS\_INTERRUPT\_RESTORE

```
#define VTSS_OS_INTERRUPT_RESTORE(
    flags ) NOT_NEEDED
```

Enable interrupts.

Definition at line 280 of file vtss\_os\_ecos.h.

## 6.25.3 Typedef Documentation

### 6.25.3.1 vtss\_mtimer\_t

```
typedef cyg_tick_count_t vtss_mtimer_t
```

Timer

Definition at line 53 of file vtss\_os\_ecos.h.

## 6.25.4 Function Documentation

### 6.25.4.1 llabs()

```
long long int llabs (
    long long int val )
```

Obtain the absolute value of a long long integer.

**Parameters**

<i>val</i>	[IN] The value to convert to absolute value.
------------	--

**Returns**

The absolute value of val.

**6.25.4.2 vtss\_callout\_malloc()**

```
void* vtss_callout_malloc (
    size_t size,
    vtss_mem_flags_t flags )
```

Callout to allocate memory.

[IN]/[OUT] seen from called function.

**Parameters**

<i>size</i>	[IN] Number of bytes to allocate.
<i>flags</i>	[IN] See vtss_mem_flags_t for details.

**Returns**

Pointer to allocated area.

**6.25.4.3 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 vtss_mem_flags_t for details.

## 6.26 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 <cctype.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()

#### 6.26.1 Detailed Description

Linux OS API.

This header file describes OS functions for Linux

## 6.26.2 Macro Definition Documentation

### 6.26.2.1 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.

### 6.26.2.2 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.

### 6.26.2.3 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.

#### 6.26.2.4 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.

#### 6.26.2.5 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.

#### 6.26.2.6 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.

### 6.26.2.7 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.

### 6.26.2.8 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.

### 6.26.2.9 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.

### 6.26.2.10 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.

### 6.26.2.11 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.

### 6.26.2.12 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.

### 6.26.2.13 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.

### 6.26.2.14 VTSS\_LABS

```
#define VTSS_LABS( arg ) labs(arg)
```

VTSS\_LABS - perform abs() on long

Definition at line 153 of file vtss\_os\_linux.h.

### 6.26.2.15 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.

### 6.26.2.16 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.

**6.26.2.17 VTSS\_OS\_CTZ64**

```
#define VTSS_OS_CTZ64( \
    val64 )
```

**Value:**

```
(({ \
    u32 _r = VTSS_OS_CTZ((u32)(val64)); \
    (val64) == 0 ? 64 : \
    _r < 32 ? _r : 32 + VTSS_OS_CTZ((u32)((val64) >> 32)); \
}))
```

Count trailing zeros of a 64-bit unsigned. Requirements/examples: `VTSS_OS_CTZ64(0x00000000_00000001) = 0` `VTSS_OS_CTZ64(0x00000000_80000000) = 31` `VTSS_OS_CTZ64(0x00000001_00000000) = 32` `VTSS_OS_CTZ64(0x80000000_00000000) = 63` `VTSS_OS_CTZ64(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.

**6.26.2.18 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.

### 6.26.2.19 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.

### 6.26.2.20 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.

## 6.27 vtss\_api/include/vtss\_otn\_api.h File Reference

OTN API.

```
#include <vtss/api/types.h>
```

### 6.27.1 Detailed Description

OTN API.

## 6.28 vtss\_api/include/vtss\_packet\_api.h File Reference

Packet API.

```
#include <vtss/api/types.h>  
#include <vtss_12_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\_JR2\_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 28

## 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)**

- **`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)
 

*Set packet configuration for port.*
- **`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.*
- **`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.*
- **`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.*

### 6.28.1 Detailed Description

Packet API.

This header file describes CPU Rx/Tx packet functions.

### 6.28.2 Macro Definition Documentation

#### 6.28.2.1 VTSS\_PRIO\_SUPER

```
#define VTSS_PRIO_SUPER VTSS_PRIO_END
```

Super priority

Definition at line 429 of file vtss\_packet\_api.h.

#### 6.28.2.2 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.

#### 6.28.2.3 VTSS\_VSTAX\_HDR\_SIZE

```
#define VTSS_VSTAX_HDR_SIZE 12
```

VStaX header size

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

#### 6.28.2.4 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.

#### 6.28.2.5 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.

#### 6.28.2.6 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.

#### 6.28.2.7 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.

#### 6.28.2.8 VTSS\_PACKET\_HDR\_SIZE\_BYTES

```
#define VTSS_PACKET_HDR_SIZE_BYTES VTSS_JR2_PACKET_HDR_SIZE_BYTES
```

Maximum header size. This define is only useful if you only compile for one target.

Definition at line 703 of file vtss\_packet\_api.h.

### 6.28.2.9 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.

### 6.28.2.10 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.

### 6.28.2.11 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.

### 6.28.2.12 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.

### 6.28.2.13 VTSS\_PACKET\_TX\_IFH\_MAX

```
#define VTSS_PACKET_TX_IFH_MAX 28
```

Tx IFH byte length (Constant)

Definition at line 2054 of file vtss\_packet\_api.h.

## 6.28.3 Enumeration Type Documentation

### 6.28.3.1 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.

## 6.28.3.2 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.

## 6.28.3.3 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.

#### 6.28.3.4 vtss\_packet\_ptp\_action\_t

enum [vtss\\_packet\\_ptp\\_action\\_t](#)

PTP actions used when encoding an injection header.

Enumerator

VTSS_PACKET_PTP_ACTION_NONE	No-op
VTSS_PACKET_PTP_ACTION_ONE_STEP	One-step PTP
VTSS_PACKET_PTP_ACTION_TWO_STEP	Two-step PTP
VTSS_PACKET_PTP_ACTION_ONE_AND_TWO_STEP	Both one- and two-step PTP
VTSS_PACKET_PTP_ACTION_ORIGIN_TIMESTAMPAMP	Update time-of-day in PTP frame's originTimestamp field

Definition at line 543 of file vtss\_packet\_api.h.

#### 6.28.3.5 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.

#### 6.28.3.6 vtss\_packet\_rx\_hints\_t

enum [vtss\\_packet\\_rx\\_hints\\_t](#)

Provides additional info on decoded extraction header.

In some cases, a frame received by the CPU should be dropped. This cannot always be detected by just looking at the decoded extraction header.

This enum provides the caller of [vtss\\_packet\\_rx\\_hdr\\_decode\(\)](#) with a means to determine whether to drop or forward the frame further to the application. Whether to actually drop it is fully up to the application.

The individual enums may be combined into a mask.

#### Enumerator

VTSS_PACKET_RX_HINTS_NONE	No hints.
VTSS_PACKET_RX_HINTS_VLAN_TAG_MISMATCH TCH	<p>If a frame is received on a C-port with a "foreign" tag (i.e. an S-tag or S-custom-tag), the frame should possibly be dropped. This is indicated with this enum being member of the hints flags.</p> <p>The same goes for frames received on S-ports or S-custom-ports with "foreign" tags.</p> <p><b>Can be set by:</b></p> <p>Luton26: Y Jaguar1: Y Serval: Y Jaguar2: Y Serval2: Y ServalT: Y</p>
VTSS_PACKET_RX_HINTS_VLAN_FRAME_MISMATCH	<p>If a tagged frame is received on a port that only should accept untagged frames or if an untagged frame is received on a port that should only accept tagged frames, then this will be set. In general, the application should not drop such frames, because e.g. BPDUs are normally untagged.</p> <p><b>Can be set by:</b></p> <p>Luton26: Y Jaguar1: Y Serval: Y Jaguar2: Y Serval2: Y ServalT: Y</p>
VTSS_PACKET_RX_HINTS_VID_MISMATCH	<p>If a frame gets classified to a VLAN ID on a port that is not member of that VID, the hints flags include this enum.</p> <p><b>Can be set by:</b></p> <p>Luton26: Y Jaguar1: Y Serval: Y Jaguar2: Y Serval2: Y ServalT: Y</p>

Definition at line 915 of file vtss\_packet\_api.h.

#### 6.28.3.7 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.

## 6.28.4 Function Documentation

### 6.28.4.1 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.

### 6.28.4.2 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.

**6.28.4.3 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.

**6.28.4.4 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.

**6.28.4.5 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.

**6.28.4.6 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.

**6.28.4.7 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.

**6.28.4.8 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.

**6.28.4.9 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.

**6.28.4.10 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.

**6.28.4.11 vtss\_packet\_tx\_frame\_port\_vlan()**

```
vtss_rc vtss_packet_tx_frame_port_vlan (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_vid_t vid,
    const u8 *const frame,
    const u32 length )
```

Send frame on port using egress rules.

**Parameters**

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>vid</i>	[IN] VLAN ID inserted if the frame is tagged on egress.
<i>frame</i>	[IN] Frame buffer excluding room for CRC.
<i>length</i>	[IN] Frame length excluding CRC.

**Returns**

Return code.

**6.28.4.12 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.

**6.28.4.13 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.
-------------	--------------------------

**6.28.4.14 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.

**6.28.4.15 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.

**6.28.4.16 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.

#### 6.28.4.17 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.

#### 6.28.4.18 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.

#### 6.28.4.19 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.

#### 6.28.4.20 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.

#### 6.28.4.21 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.

#### 6.28.4.22 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.

**6.28.4.23 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.

**6.28.4.24 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.

**6.28.4.25 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.

**6.28.4.26 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.

#### 6.28.4.27 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.

## 6.29 vtss\_api/include/vtss\_pcs\_10gbase\_r\_api.h File Reference

PCS\_10BASE\_R API.

```
#include <vtss/api/types.h>
```

### 6.29.1 Detailed Description

PCS\_10BASE\_R API.

## 6.30 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 KR Training & Autoneg config*
- struct [vtss\\_phy\\_10g\\_kr\\_status\\_aneg\\_t](#)  
*10G Phy Base KR Autoneg status*
- struct [vtss\\_phy\\_10g\\_kr\\_status\\_train\\_t](#)  
*10G Phy Base KR Training 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](#)
- 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](#)
- struct [vtss\\_phy\\_10g\\_pcs\\_prbs\\_gen\\_conf\\_t](#)
- struct [vtss\\_phy\\_10g\\_pcs\\_prbs\\_mon\\_conf\\_t](#)
- struct [vtss\\_phy\\_10g\\_prbs\\_gen\\_conf\\_t](#)
- struct [vtss\\_phy\\_10g\\_prbs\\_mon\\_conf\\_t](#)  
*10G PHY prbs monitor Configuration*
- struct [vtss\\_phy\\_10g\\_pkt\\_gen\\_conf\\_t](#)  
*10G PHY Packet generator configuration*
- struct [vtss\\_phy\\_10g\\_pkt\\_mon\\_conf\\_t](#)  
*10G PHY Packet Monitor configuration*
- struct [vtss\\_phy\\_10g\\_timestamp\\_val\\_t](#)  
*10G PHY timestamp value array(holder)*
- struct [vtss\\_phy\\_10g\\_id\\_t](#)  
*10G Phy part number and revision*
- struct [vtss\\_gpio\\_10g\\_gpio\\_mode\\_t](#)  
*GPIO configured mode.*
- struct [vtss\\_phy\\_10g\\_fw\\_status\\_t](#)  
*Firmware status.*

## Macros

- #define VTSS\_SLEWRATE\_25PS 0  
*Slew rate ctrl of OB.*
- #define VTSS\_SLEWRATE\_35PS 1
- #define VTSS\_SLEWRATE\_55PS 2
- #define VTSS\_SLEWRATE\_70PS 3
- #define VTSS\_SLEWRATE\_120PS 4
- #define VTSS\_SLEWRATE\_INVALID 5
- #define BOOLEAN\_STORAGE\_COUNT 6
- #define UNSIGNED\_STORAGE\_COUNT 5
- #define PHASE\_POINTS 128
- #define AMPLITUDE\_POINTS 64
- #define VTSS\_PHY\_10G\_ONE\_LINE\_ACTIVE 0x08
- #define VTSS\_PHY\_10G\_MACSEC\_DISABLED 0x04
- #define VTSS\_PHY\_10G\_TIMESTAMP\_DISABLED 0x02
- #define VTSS\_PHY\_10G\_MACSEC\_KEY\_128 0x01
- #define VTSS\_10G\_PHY\_GPIO\_MAX 12
- #define VTSS\_10G\_PHY\_GPIO\_MAL\_MAX 40
- #define VTSS\_PHY\_10G\_LINK\_LOS\_EV 0x00000001  
*Event source identification mask values.*
- #define VTSS\_PHY\_10G\_RX\_LOL\_EV 0x00000002
- #define VTSS\_PHY\_10G\_TX\_LOL\_EV 0x00000004
- #define VTSS\_PHY\_10G\_LOPC\_EV 0x00000008
- #define VTSS\_PHY\_10G\_HIGH\_BER\_EV 0x00000010
- #define VTSS\_PHY\_10G\_MODULE\_STAT\_EV 0x00000020
- #define VTSS\_PHY\_10G\_PCS\_RECEIVE\_FAULT\_EV 0x00000040
- #define VTSS\_PHY\_EWIS\_SEF\_EV 0x00000080
- #define VTSS\_PHY\_EWIS\_FPLM\_EV 0x00000100
- #define VTSS\_PHY\_EWIS\_FAIS\_EV 0x00000200
- #define VTSS\_PHY\_EWIS\_LOF\_EV 0x00000400
- #define VTSS\_PHY\_EWIS\_RDIL\_EV 0x00000800
- #define VTSS\_PHY\_EWIS\_AISL\_EV 0x00001000
- #define VTSS\_PHY\_EWIS\_LCDP\_EV 0x00002000
- #define VTSS\_PHY\_EWIS\_PLMP\_EV 0x00004000
- #define VTSS\_PHY\_EWIS\_AISP\_EV 0x00008000
- #define VTSS\_PHY\_EWIS\_LOPP\_EV 0x00010000
- #define VTSS\_PHY\_EWIS\_UNEQP\_EV 0x00020000
- #define VTSS\_PHY\_EWIS\_FEUNEQP\_EV 0x00040000
- #define VTSS\_PHY\_EWIS\_FERDIP\_EV 0x00080000
- #define VTSS\_PHY\_EWIS\_REIL\_EV 0x00100000
- #define VTSS\_PHY\_EWIS\_REIP\_EV 0x00200000
- #define VTSS\_PHY\_EWIS\_B1\_NZ\_EV 0x00400000
- #define VTSS\_PHY\_EWIS\_B2\_NZ\_EV 0x00800000
- #define VTSS\_PHY\_EWIS\_B3\_NZ\_EV 0x01000000
- #define VTSS\_PHY\_EWIS\_REIL\_NZ\_EV 0x02000000
- #define VTSS\_PHY\_EWIS\_REIP\_NZ\_EV 0x04000000
- #define VTSS\_PHY\_EWIS\_B1\_THRESH\_EV 0x08000000
- #define VTSS\_PHY\_EWIS\_B2\_THRESH\_EV 0x10000000
- #define VTSS\_PHY\_EWIS\_B3\_THRESH\_EV 0x20000000
- #define VTSS\_PHY\_EWIS\_REIL\_THRESH\_EV 0x40000000
- #define VTSS\_PHY\_EWIS\_REIP\_THRESH\_EV 0x80000000
- #define VTSS\_PHY\_10G\_RX\_LOS\_EV 0x00000001
- #define VTSS\_PHY\_10G\_RX\_LOL\_EV 0x00000002

- #define VTSS\_PHY\_10G\_TX\_LOL\_EV 0x00000004
- #define VTSS\_PHY\_10G\_RX\_CHAR\_DEC\_CNT\_THRESH\_EV 0x00000010
- #define VTSS\_PHY\_10G\_RX\_CHAR\_ENC\_CNT\_THRESH\_EV 0x00000020
- #define VTSS\_PHY\_10G\_RX\_BLK\_DEC\_CNT\_THRESH\_EV 0x00000040
- #define VTSS\_PHY\_10G\_RX\_BLK\_ENC\_CNT\_THRESH\_EV 0x00000080
- #define VTSS\_PHY\_10G\_RX\_SEQ\_CNT\_THRESH\_EV 0x00000100
- #define VTSS\_PHY\_10G\_RX\_SEQ\_CNT\_THRESH\_EV 0x00000200
- #define VTSS\_PHY\_10G\_FEC\_UNFIXED\_CNT\_THRESH\_EV 0x00000400
- #define VTSS\_PHY\_10G\_FEC\_FIXED\_CNT\_THRESH\_EV 0x00000800
- #define VTSS\_PHY\_10G\_HIGHBER\_EV 0x00001000
- #define VTSS\_PHY\_10G\_RX\_LINK\_STAT\_EV 0x00002000
- #define VTSS\_PHY\_10G\_RX\_LINK\_STAT\_EV 0x00002000
- #define VTSS\_PHY\_10G\_GPIO\_INT\_AGG0\_EV 0x00004000
- #define VTSS\_PHY\_10G\_GPIO\_INT\_AGG1\_EV 0x00008000
- #define VTSS\_PHY\_10G\_GPIO\_INT\_AGG2\_EV 0x00010000
- #define VTSS\_PHY\_10G\_GPIO\_INT\_AGG3\_EV 0x00020000
- #define VTSS\_PHY\_1G\_LINE\_AUTONEG\_RESTART\_EV 0x00040000
- #define VTSS\_PHY\_1G\_HOST\_AUTONEG\_RESTART\_EV 0x00080000
- #define VTSS\_PHY\_10G\_LINE\_MAC\_LOCAL\_FAULT\_EV 0x00100000
- #define VTSS\_PHY\_10G\_HOST\_MAC\_LOCAL\_FAULT\_EV 0x00400000
- #define VTSS\_PHY\_10G\_LINE\_MAC\_REMOTE\_FAULT\_EV 0x00800000
- #define VTSS\_PHY\_10G\_HOST\_MAC\_REMOTE\_FAULT\_EV 0x01000000

## Typedefs

- typedef u16 vtss\_gpio\_10g\_no\_t
- typedef enum ckout\_sel\_ckout\_sel\_t
 

*10G Phy CKOUTs Enum*
- typedef u32 vtss\_32\_cntr\_t
- typedef u32 vtss\_phy\_10g\_event\_t
- typedef u32 vtss\_phy\_10g\_extnd\_event\_t

## Enumerations

- enum oper\_mode\_t {
 VTSS\_PHY\_LAN\_MODE, VTSS\_PHY\_WAN\_MODE, VTSS\_PHY\_1G\_MODE, VTSS\_PHY\_LAN\_SYNCE\_MODE,
 VTSS\_PHY\_WAN\_SYNCE\_MODE, VTSS\_PHY\_LAN\_MIXED\_SYNCE\_MODE, VTSS\_PHY\_WAN\_MIXED\_SYNCE\_MODE,
 VTSS\_PHY\_REPEATERO\_MODE }
 

*10G Phy operating mode enum type*
- enum vtss\_wrefclk\_t { VTSS\_WREFCLK\_155\_52, VTSS\_WREFCLK\_622\_08 }
 

*Modes for WAN reference clock.*
- enum vtss\_phy\_interface\_mode {
 VTSS\_PHY\_XAUI\_XFI, VTSS\_PHY\_XGMII\_XFI, VTSS\_PHY\_RXAUI\_XFI, VTSS\_PHY\_SGMII\_LANE\_0\_XFI,
 VTSS\_PHY\_SGMII\_LANE\_3\_XFI, VTSS\_PHY\_SFI\_XFI }
 

*Phy Interface modes.*
- enum vtss\_recvrd\_t { VTSS\_RECVRD\_RXCLKOUT, VTSS\_RECVRD\_TXCLKOUT }
 

*Modes for recovered clock.*
- enum vtss\_recvrdclk\_cdr\_div\_t { VTSS\_RECVRDCLK\_CDR\_DIV\_64, VTSS\_RECVRDCLK\_CDR\_DIV\_66 }
 

*Modes for recovered clock divisor.*
- enum vtss\_srefclk\_div\_t { VTSS\_SREFCLK\_DIV\_64, VTSS\_SREFCLK\_DIV\_66, VTSS\_SREFCLK\_DIV\_16 }
 

*Modes for Synch-E recovered clock.*

- enum `vtss_wref_clk_div_t` { `VTSS_WREFCLK_NONE`, `VTSS_WREFCLK_DIV_16` }
 

*Modes for WREFCLK clock divisor.*
- enum `apc_ib_regulator_t` { `VTSS_AP_C_IB_SFP_PLUS_ZR`, `VTSS_AP_C_IB_BACKPLANE` }
 

*APC Rx regulator mode.*
- enum `ddr_mode_t` { `VTSS_DDR_MODE_A`, `VTSS_DDR_MODE_K`, `VTSS_DDR_MODE_M` }
 

*Interleave mode.*
- enum `clk_mstr_t` { `VTSS_CLK_MSTR_INTERNAL`, `VTSS_CLK_MSTR_EXTERNAL` }
 

*Clock master.*
- enum `vtss_rptr_rate_t` {
 `VTSS_RPTR_RATE_NONE`, `VTSS_RPTR_RATE_10_3125`, `VTSS_RPTR_RATE_9_9532`, `VTSS_RPTR_RATE_11_3`,  
`VTSS_RPTR_RATE_10_5187`, `VTSS_RPTR_RATE_1_25`, `VTSS_RPTR_RATE_10_709`, `VTSS_RPTR_RATE_11_095727`,  
`VTSS_RPTR_RATE_11_05` }
 

*Repeater Data rate.*
- enum `vtss_phy_10g_media_t` {
 `VTSS_MEDIA_TYPE_SR`, `VTSS_MEDIA_TYPE_SR2`, `VTSS_MEDIA_TYPE_DAC`, `VTSS_MEDIA_TYPE_ZR`,  
`VTSS_MEDIA_TYPE_KR`, `VTSS_MEDIA_TYPE_SR_SC`, `VTSS_MEDIA_TYPE_SR2_SC`, `VTSS_MEDIA_TYPE_DAC_SC`,  
`VTSS_MEDIA_TYPE_ZR_SC`, `VTSS_MEDIA_TYPE_ZR2_SC`, `VTSS_MEDIA_TYPE_KR_SC`, `VTSS_MEDIA_TYPE_NONE` }
 

*10G Phy Media type*
- enum `vtss_phy_6g_link_partner_distance_t` { `VTSS_6G_LINK_SHORT_RANGE`, `VTSS_6G_LINK_LONG_RANGE` }
 

*6G serdes link partner distance selection*
- enum `vtss_phy_10g_ib_apc_op_mode_t` {
 `VTSS_IB_AP_C_AUTO`, `VTSS_IB_AP_C_MANUAL`, `VTSS_IB_AP_C_FREEZE`, `VTSS_IB_AP_C_RESET`,  
`VTSS_IB_AP_C_RESTART`, `VTSS_IB_AP_C_NONE` }
 

*10G SERDES APC operation*
- enum `vtss_channel_t` {
 `VTSS_CHANNEL_AUTO`, `VTSS_CHANNEL_0`, `VTSS_CHANNEL_1`, `VTSS_CHANNEL_2`,  
`VTSS_CHANNEL_3` }
 

*Channel modes - Auto is recommended.*
- enum `vtss_reccrd_clkout_t` { `VTSS_RECVRD_CLKOUT_DISABLE`, `VTSS_RECVRD_CLKOUT_LINE_SIDE_RX_CLOCK`,  
`VTSS_RECVRD_CLKOUT_LINE_SIDE_TX_CLOCK` }
 

*Modes for (rx/tx) recovered clock output.*
- enum `vtss_phy_10g_srefclk_freq_t` { `VTSS_PHY_10G_SREFCLK_156_25`, `VTSS_PHY_10G_SREFCLK_125_00`,  
`VTSS_PHY_10G_SREFCLK_155_52`, `VTSS_PHY_10G_SREFCLK_INVALID` }
 

*10G Phy sref clock input frequency*
- enum `vtss_phy_10g_ckout_freq_t` { `VTSS_PHY_10G_CLK_FULL_RATE`, `VTSS_PHY_10G_CLK_DIVIDE_BY_2`,  
`VTSS_PHY_10G_CLK_INVALID` }
 

*10G Phy clock frequency*
- enum `vtss_ckout_data_sel_t` {
 `VTSS_CKOUT_LINE0_TX_CLOCK`, `VTSS_CKOUT_LINE1_TX_CLOCK`, `VTSS_CKOUT_LINE2_TX_CLOCK`,  
`VTSS_CKOUT_LINE3_TX_CLOCK`,  
`VTSS_CKOUT_HOST0_TX_CLOCK`, `VTSS_CKOUT_HOST1_TX_CLOCK`, `VTSS_CKOUT_HOST2_TX_CLOCK`,  
`VTSS_CKOUT_HOST3_TX_CLOCK`,  
`VTSS_CKOUT_LINE0_RECVRD_CLOCK`, `VTSS_CKOUT_LINE1_RECVRD_CLOCK`, `VTSS_CKOUT_LINE2_RECVRD_CLOCK`,  
`VTSS_CKOUT_LINE3_RECVRD_CLOCK`,  
`VTSS_CKOUT_HOST0_RECVRD_CLOCK`, `VTSS_CKOUT_HOST1_RECVRD_CLOCK`, `VTSS_CKOUT_HOST2_RECVRD_CLOCK`,  
`VTSS_CKOUT_HOST3_RECVRD_CLOCK`,  
`VTSS_CKOUT_HOST_PLL_CLOCK`, `VTSS_CKOUT_LINE_PLL_CLOCK`, `VTSS_CKOUT_CSR_CLOCK`,  
`VTSS_CKOUT_LTC_CLOCK`,  
`VTSS_CKOUT_DF2F_CLOCK`, `VTSS_CKOUT_F2DF_CLOCK`, `VTSS_CKOUT_DEBUG1`, `VTSS_CKOUT_DEBUG2`,  
`VTSS_CKOUT_OSCILLATOR_OUTPUT` }
 

*Modes for recovered clock output.*

- enum `vtss_phy_10g_squelch_src_t` {
   
VTSS\_CKOUT\_SQUELCH\_SRC\_GPIO0, VTSS\_CKOUT\_SQUELCH\_SRC\_GPIO1, VTSS\_CKOUT\_SQUELCH\_SRC\_GPIO2,
   
VTSS\_CKOUT\_SQUELCH\_SRC\_GPIO3,
   
VTSS\_CKOUT\_SQUELCH\_SRC\_GPIO4, VTSS\_CKOUT\_SQUELCH\_SRC\_GPIO5, VTSS\_CKOUT\_SQUELCH\_SRC\_GPIO6,
   
VTSS\_CKOUT\_SQUELCH\_SRC\_GPIO7,
   
VTSS\_CKOUT\_SQUELCH\_SRC\_LINK\_LINE0, VTSS\_CKOUT\_SQUELCH\_SRC\_LINK\_LINE1, VTSS\_CKOUT\_SQUELCH\_SRC\_LINK\_LINE2,
   
VTSS\_CKOUT\_SQUELCH\_SRC\_LINK\_LINE3,
   
VTSS\_CKOUT\_SQUELCH\_SRC\_LINK\_HOST0, VTSS\_CKOUT\_SQUELCH\_SRC\_LINK\_HOST1, VTSS\_CKOUT\_SQUELCH\_SRC\_LINK\_HOST2,
   
VTSS\_CKOUT\_SQUELCH\_SRC\_LINK\_HOST3,
   
VTSS\_CKOUT\_SQUELCH\_SRC\_LOS\_LINE0, VTSS\_CKOUT\_SQUELCH\_SRC\_LOS\_LINE1, VTSS\_CKOUT\_SQUELCH\_SRC\_LOS\_LINE2,
   
VTSS\_CKOUT\_SQUELCH\_SRC\_LOS\_LINE3,
   
VTSS\_CKOUT\_SQUELCH\_SRC\_LOS\_HOST0, VTSS\_CKOUT\_SQUELCH\_SRC\_LOS\_HOST1, VTSS\_CKOUT\_SQUELCH\_SRC\_LOS\_HOST2,
   
VTSS\_CKOUT\_SQUELCH\_SRC\_LOS\_HOST3,
   
VTSS\_CKOUT\_SQUELCH\_SRC\_LINK\_LINE0\_KR, VTSS\_CKOUT\_SQUELCH\_SRC\_LINK\_LINE1\_KR,
   
VTSS\_CKOUT\_SQUELCH\_SRC\_LINK\_LINE2\_KR, VTSS\_CKOUT\_SQUELCH\_SRC\_LINK\_LINE3\_KR,
   
VTSS\_CKOUT\_SQUELCH\_SRC\_LINK\_HOST0\_KR, VTSS\_CKOUT\_SQUELCH\_SRC\_LINK\_HOST1\_KR,
   
VTSS\_CKOUT\_SQUELCH\_SRC\_LINK\_HOST2\_KR, VTSS\_CKOUT\_SQUELCH\_SRC\_LINK\_HOST3\_KR,
   
VTSS\_CKOUT\_NO\_SQUELCH }
   
*squelch control source*
- enum `vtss_phy_10g_clk_sel_t` {
   
VTSS\_PHY\_10G\_LINE0\_RECVRD\_CLOCK, VTSS\_PHY\_10G\_LINE1\_RECVRD\_CLOCK, VTSS\_PHY\_10G\_LINE2\_RECVRD\_CLOCK,
   
VTSS\_PHY\_10G\_LINE3\_RECVRD\_CLOCK,
   
VTSS\_PHY\_10G\_HOST0\_RECVRD\_CLOCK, VTSS\_PHY\_10G\_HOST1\_RECVRD\_CLOCK, VTSS\_PHY\_10G\_HOST2\_RECVRD\_CLOCK,
   
VTSS\_PHY\_10G\_HOST3\_RECVRD\_CLOCK,
   
VTSS\_PHY\_10G\_SREFCLK, VTSS\_PHY\_10G\_SYNC\_DISABLE = 15 }
   
*Modes of recovered clocks for ckout and sckout pins.*
- enum `vtss_phy_10g_recvrd_clk_sel_t` {
   
VTSS\_PHY\_10G\_USE\_LINE0\_RECVRD\_CLOCK, VTSS\_PHY\_10G\_USE\_LINE1\_RECVRD\_CLOCK,
   
VTSS\_PHY\_10G\_USE\_LINE2\_RECVRD\_CLOCK, VTSS\_PHY\_10G\_USE\_LINE3\_RECVRD\_CLOCK,
   
VTSS\_PHY\_10G\_USE\_HOST0\_RECVRD\_CLOCK, VTSS\_PHY\_10G\_USE\_HOST1\_RECVRD\_CLOCK,
   
VTSS\_PHY\_10G\_USE\_HOST2\_RECVRD\_CLOCK, VTSS\_PHY\_10G\_USE\_HOST3\_RECVRD\_CLOCK,
   
VTSS\_PHY\_10G\_USE\_SREFCLK\_CLOCK, VTSS\_PHY\_10G\_USE\_DEFAULT\_RECVRD\_CLOCK }
   
*Modes of recovered clock selection.*
- enum `ckout_sel_t` { **CKOUT0, CKOUT1, CKOUT2, CKOUT3** }
   
*10G Phy CKOUTs Enum*
- enum `vtss_phy_10g_sckout_freq_t` { VTSS\_PHY\_10G\_SCKOUT\_156\_25, VTSS\_PHY\_10G\_SCKOUT\_125\_00,
   
VTSS\_PHY\_10G\_SCKOUT\_INVALID }
   
*10G Phy sckout clock input frequency*
- enum `vtss_phy_10g_rx_macro_t` { VTSS\_PHY\_10G\_RX\_MACRO\_LINE, VTSS\_PHY\_10G\_RX\_MACRO\_HOST,
   
VTSS\_PHY\_10G\_RX\_MACRO\_SREFCLK }
   
*10G Phy Rx MACRO Configuration*
- enum `vtss_phy_10g_tx_macro_t` { VTSS\_PHY\_10G\_TX\_MACRO\_LINE, VTSS\_PHY\_10G\_TX\_MACRO\_HOST,
   
VTSS\_PHY\_10G\_TX\_MACRO\_SCKOUT }
   
*10G Phy tx MACRO Configuration*
- enum `vtss_phy_10g_clause_37_remote_fault_t` { VTSS\_PHY\_10G\_CLAUSE\_37\_RF\_LINK\_OK, VTSS\_PHY\_10G\_CLAUSE\_37\_RF\_LINK\_FAILURE,
   
VTSS\_PHY\_10G\_CLAUSE\_37\_RF\_LINK\_FAILURE, VTSS\_PHY\_10G\_CLAUSE\_37\_RF\_AUTONEG\_ERROR }
   
*Auto-negotiation remote fault type.*
- enum `vtss_lb_type_t` {
   
VTSS\_LB\_NONE, VTSS\_LB\_SYSTEM\_XS\_SHALLOW, VTSS\_LB\_SYSTEM\_XS\_DEEP, VTSS\_LB\_SYSTEM\_PCS\_SHALLOW,
   
VTSS\_LB\_SYSTEM\_PCS\_DEEP, VTSS\_LB\_SYSTEM\_PMA, VTSS\_LB\_NETWORK\_XS\_SHALLOW,
   
VTSS\_LB\_NETWORK\_XS\_DEEP,
   
VTSS\_LB\_NETWORK\_PCS, VTSS\_LB\_NETWORK\_WIS, VTSS\_LB\_NETWORK\_PMA, VTSS\_LB\_H2,
   
VTSS\_LB\_H3, VTSS\_LB\_H4, VTSS\_LB\_H5, VTSS\_LB\_H6,
   
VTSS\_LB\_L0, VTSS\_LB\_L1, VTSS\_LB\_L2, VTSS\_LB\_L3,
   
**VTSS\_LB\_L2C** }
   
*Link layer type*

- *10G loopback types*
- enum `vtss_phy_10g_power_t` { `VTSS_PHY_10G_POWER_ENABLE`, `VTSS_PHY_10G_POWER_DISABLE` }
- 10G Phy power setting*
- enum `vtss_phy_10g_failover_mode_t` {  
`VTSS_PHY_10G_PMA_TO_FROM_XAUI_NORMAL`, `VTSS_PHY_10G_PMA_TO_FROM_XAUI_CROSSED`,  
`VTSS_PHY_10G_PMA_0_TO_FROM_XAUI_0_TO_XAUI_1`, `VTSS_PHY_10G_PMA_0_TO_FROM_XAUI_1_TO_XAUI_0`,  
`VTSS_PHY_10G_PMA_1_TO_FROM_XAUI_0_TO_XAUI_1`, `VTSS_PHY_10G_PMA_1_TO_FROM_XAUI_1_TO_XAUI_0` }
- 10G Phy Failover Mode Setting*
- enum `vtss_phy_10g_auto_failover_event_t` {  
`VTSS_PHY_10G_AUTO_FAILOVER_EVENT_PCS_LINK_STATUS`, `VTSS_PHY_10G_AUTO_FAILOVER_EVENT_SERDES`,  
`VTSS_PHY_10G_AUTO_FAILOVER_EVENT_WIS_LOF`, `VTSS_PHY_10G_AUTO_FAILOVER_EVENT_GPIO`,  
`VTSS_PHY_10G_AUTO_FAILOVER_EVENT_NONE` }
- 10G Phy Automatic Failover Event Setting*
- enum `vtss_phy_10g_auto_failover_filter_t` {  
`VTSS_PHY_10G_AUTO_FAILOVER_FILTER_NONE`, `VTSS_PHY_10G_AUTO_FAILOVER_FILTER_CNT_B2316`,  
`VTSS_PHY_10G_AUTO_FAILOVER_FILTER_CNT_B70`, `VTSS_PHY_10G_AUTO_FAILOVER_FILTER_CNT_A2316`,  
`VTSS_PHY_10G_AUTO_FAILOVER_FILTER_CNT_A70` }
- 10G PHY Automatic Failover Filter*
- enum `vtss_phy_10g_vscope_scan_t` { `VTSS_PHY_10G_FAST_SCAN`, `VTSS_PHY_10G_FAST_SCAN_PLUS`,  
`VTSS_PHY_10G_QUICK_SCAN`, `VTSS_PHY_10G_FULL_SCAN` }
- VSCOPE scan types.*
- enum `vtss_phy_10g_pkt_mon_rst_t` {  
`VTSS_PHY_10G_PKT_MON_RST_ALL`, `VTSS_PHY_10G_PKT_MON_RST_GOOD`, `VTSS_PHY_10G_PKT_MON_RST_BAD`,  
`VTSS_PHY_10G_PKT_MON_RST_FRAG`,  
`VTSS_PHY_10G_PKT_MON_RST_LFAULT`, `VTSS_PHY_10G_PKT_MON_RST_BER`, `VTSS_PHY_10G_PKT_MON_RST_NOMATCH` }
- 10G PHY Packet monitor configuration*
- enum `vtss_phy_10g_type_t` {  
`VTSS_PHY_TYPE_10G_NONE` = 0, `VTSS_PHY_TYPE_8484` = 8484, `VTSS_PHY_TYPE_8486` = 8486,  
`VTSS_PHY_TYPE_8487` = 8487,  
`VTSS_PHY_TYPE_8488` = 8488, `VTSS_PHY_TYPE_8489` = 8489, `VTSS_PHY_TYPE_8489_15` = 848915,  
`VTSS_PHY_TYPE_8490` = 8490,  
`VTSS_PHY_TYPE_8491` = 8491, `VTSS_PHY_TYPE_8256` = 8256, `VTSS_PHY_TYPE_8257` = 8257, `VTSS_PHY_TYPE_8258` = 8258,  
`VTSS_PHY_TYPE_8254` = 8254 }
- 10g PHY type*
- enum `vtss_phy_10g_family_t` {  
`VTSS_PHY_FAMILY_10G_NONE`, `VTSS_PHY_FAMILY_XAUI_XGMII_XFI`, `VTSS_PHY_FAMILY_XAU_I_XFI`,  
`VTSS_PHY_FAMILY_VENICE`,  
`VTSS_PHY_FAMILY_MALIBU` }
- 10G PHY family*
- enum `vtss_10g_phy_gpio_t` {  
`VTSS_10G_PHY_GPIO_NOT_INITIALIZED`, `VTSS_10G_PHY_GPIO_OUT`, `VTSS_10G_PHY_GPIO_IN`,  
`VTSS_10G_PHY_GPIO_WIS_INT`,  
`VTSS_10G_PHY_GPIO_1588_LOAD_SAVE`, `VTSS_10G_PHY_GPIO_1588_1PPS_0`, `VTSS_10G_PHY_GPIO_1588_1PPS_1`,  
`VTSS_10G_PHY_GPIO_1588_1PPS_2`,  
`VTSS_10G_PHY_GPIO_1588_1PPS_3`, `VTSS_10G_PHY_GPIO_PCS_RX_FAULT`, `VTSS_10G_PHY_GPIO_SET_I2C_MASTER`,  
`VTSS_10G_PHY_GPIO_TX_ENABLE`,  
`VTSS_10G_PHY_GPIO_LINE_PLL_STATUS`, `VTSS_10G_PHY_GPIO_HOST_PLL_STATUS`, `VTSS_10G_PHY_GPIO_RCO`,  
`VTSS_10G_PHY_GPIO_CHAN_INT_0`,  
`VTSS_10G_PHY_GPIO_CHAN_INT_1`, `VTSS_10G_PHY_GPIO_1588_INT`, `VTSS_10G_PHY_GPIO_TS_FIFO_EMPTY`,  
`VTSS_10G_PHY_GPIO_AGG_INT_0`,  
`VTSS_10G_PHY_GPIO_AGG_INT_1`, `VTSS_10G_PHY_GPIO_AGG_INT_2`, `VTSS_10G_PHY_GPIO_AGG_INT_3`,  
`VTSS_10G_PHY_GPIO_PLL_INT_0`,

```
VTSS_10G_PHY_GPIO_PLL_INT_1, VTSS_10G_PHY_GPIO_SET_I2C_SLAVE, VTSS_10G_PHY_GPIO_CRSS_INT,
VTSS_10G_PHY_GPIO_LED,
VTSS_10G_PHY_GPIO_DRIVE_LOW, VTSS_10G_PHY_GPIO_DRIVE_HIGH }
```

*GPIO configured mode.*

- enum `vtss_gpio_10g_gpio_intr_sgnl_t` {
 

```
VTSS_10G_GPIO_INTR_SGNL_I2C_MSTR_DATA_OUT, VTSS_10G_GPIO_INTR_SGNL_I2C_MSTR_CLK_OUT,
VTSS_10G_GPIO_INTR_SGNL_LED_TX, VTSS_10G_GPIO_INTR_SGNL_LED_RX,
VTSS_10G_GPIO_INTR_SGNL_RX_ALARM, VTSS_10G_GPIO_INTR_SGNL_TX_ALARM, VTSS_10G_GPIO_INTR_SGNL_
VTSS_10G_GPIO_INTR_SGNL_LINE_LINK,
VTSS_10G_GPIO_INTR_SGNL_LINE_KR_8b10b_2GPIO, VTSS_10G_GPIO_INTR_SGNL_LINE_KR_10b_2GPIO,
VTSS_10G_GPIO_INTR_SGNL_ROSI_PULSE, VTSS_10G_GPIO_INTR_SGNL_ROSI_SDATA,
VTSS_10G_GPIO_INTR_SGNL_ROSI_SCLK, VTSS_10G_GPIO_INTR_SGNL_TOSI_PULSE, VTSS_10G_GPIO_INTR_SGNL_
VTSS_10G_GPIO_INTR_SGNL_LINE_PCS1G_LINK,
VTSS_10G_GPIO_INTR_SGNL_LINE_PCS_RX_STAT, VTSS_10G_GPIO_INTR_SGNL_CLIENT_PCS1G_LINK,
VTSS_10G_GPIO_INTR_SGNL_HOST_PCS_RX_STAT, VTSS_10G_GPIO_INTR_SGNL_HOST_SD10G_IB_SIG,
VTSS_10G_GPIO_INTR_SGNL_LINE_SD10G_IB_SIG, VTSS_10G_GPIO_INTR_SGNL_HPCS_INTR,
VTSS_10G_GPIO_INTR_SGNL_LPCS_INTR, VTSS_10G_GPIO_INTR_SGNL_CLIENT_PCS1G_INTR,
VTSS_10G_GPIO_INTR_SGNL_LINE_PCS1G_INTR, VTSS_10G_GPIO_INTR_SGNL_WIS_INT0,
VTSS_10G_GPIO_INTR_SGNL_HOST_PMA_INT, VTSS_10G_GPIO_INTR_SGNL_LINE_PMA_INT,
VTSS_10G_GPIO_INTR_SGNL_DATA_ACT_TX, VTSS_10G_GPIO_INTR_SGNL_DATA_ACT_RX,
VTSS_10G_GPIO_INTR_SGNL_HDATA_ACT_TX, VTSS_10G_GPIO_INTR_SGNL_HDATA_ACT_RX,
VTSS_10G_GPIO_INTR_SGNL_XGMII_PAUS_EGR, VTSS_10G_GPIO_INTR_SGNL_XGMII_PAUS_ING,
VTSS_10G_GPIO_INTR_SGNL_RX_PCS_PAUS, VTSS_10G_GPIO_INTR_SGNL_TX_PCS_PAUS,
VTSS_10G_GPIO_INTR_SGNL_RX_WIS_PAUS, VTSS_10G_GPIO_INTR_SGNL_TX_WIS_PAUS,
VTSS_10G_GPIO_INTR_SGNL_ETH_CHAN_DIS, VTSS_10G_GPIO_INTR_SGNL_MACSEC_1588_SFD_LANE,
VTSS_10G_GPIO_INTR_SGNL_LINE_S_TXFAULT, VTSS_10G_GPIO_INTR_SGNL_LPCS1G_LATENCY0_OR_EWIS_BIT,
VTSS_10G_GPIO_INTR_SGNL_LPCS1G_LATENCY1_OR_EWIS_BIT1, VTSS_10G_GPIO_INTR_SGNL_LPCS1G_CHAR,
VTSS_10G_GPIO_INTR_SGNL_LPCS1G_CHAR_POS1_OR_EWIS_WORD0, VTSS_10G_GPIO_INTR_SGNL_LPCS1G_CHAR,
VTSS_10G_GPIO_INTR_SGNL_LPCS1G_CHAR_POS3_OR_EWIS_WORD2, VTSS_10G_GPIO_INTR_SGNL_MACSEC_IC,
VTSS_10G_GPIO_INTR_SGNL_MACSEC_IGR_PRED_VAR1, VTSS_10G_GPIO_INTR_SGNL_KR_ACTV_2GPIO,
VTSS_10G_GPIO_INTR_SGNL_DFT_TX_2GPIO, VTSS_10G_GPIO_INTR_SGNL_RESERVED,
VTSS_10G_GPIO_INTR_SGNL_EXE_LST_2GPIO_0, VTSS_10G_GPIO_INTR_SGNL_EXE_LST_2GPIO_1,
VTSS_10G_GPIO_INTR_SGNL_EXE_LST_2GPIO_2, VTSS_10G_GPIO_INTR_SGNL_EXE_LST_2GPIO_3,
VTSS_10G_GPIO_INTR_SGNL_EXE_LST_2GPIO_4, VTSS_10G_GPIO_INTR_SGNL_LINK_HCD_2GPIO_0,
VTSS_10G_GPIO_INTR_SGNL_LINK_HCD_2GPIO_1, VTSS_10G_GPIO_INTR_SGNL_LINK_HCD_2GPIO_2,
VTSS_10G_GPIO_INTR_SGNL_ETH_1G_ENA, VTSS_10G_GPIO_INTR_SGNL_H_KR_8b10b_2GPIO,
VTSS_10G_GPIO_INTR_SGNL_H_KR_10Gb_2GPIO, VTSS_10G_GPIO_INTR_SGNL_H_KR_ACTV_2GPIO,
VTSS_10G_GPIO_INTR_SGNL_NONE }
```

*GPIO internal signal types.*

- enum `vtss_gpio_10g_chan_intrpt_t` {
 

```
VTSS_10G_GPIO_INTRPT_WISO, VTSS_10G_GPIO_INTRPT_WIS1, VTSS_10G_GPIO_INTRPT_LPCS10G,
VTSS_10G_GPIO_INTRPT_HPCS10G,
VTSS_10G_GPIO_INTRPT_LPCS1G, VTSS_10G_GPIO_INTRPT_HPCS1G, VTSS_10G_GPIO_INTRPT_MSEC_EGR,
VTSS_10G_GPIO_INTRPT_MSEC_IGR,
VTSS_10G_GPIO_INTRPT_LMAC, VTSS_10G_GPIO_INTRPT_HMAC, VTSS_10G_GPIO_INTRPT_FCBUF,
VTSS_10G_GPIO_INTRPT_LIGR_FIFO,
VTSS_10G_GPIO_INTRPT_LEGRT_FIFO, VTSS_10G_GPIO_INTRPT_HEGR_FIFO, VTSS_10G_GPIO_INTRPT_LPMA,
VTSS_10G_GPIO_INTRPT_HPMA }
```

*GPIO Channel level interrupts.*

- enum `vtss_gpio_10g_aggr_intrpt_t` {
 

```
VTSS_10G_GPIO_AGGR_INTRPT_CH0_INTR0_EN, VTSS_10G_GPIO_AGGR_INTRPT_CH0_INTR1_EN,
VTSS_10G_GPIO_AGGR_INTRPT_CH1_INTR0_EN, VTSS_10G_GPIO_AGGR_INTRPT_CH1_INTR1_EN,
VTSS_10G_GPIO_AGGR_INTRPT_CH2_INTR0_EN, VTSS_10G_GPIO_AGGR_INTRPT_CH2_INTR1_EN,
VTSS_10G_GPIO_AGGR_INTRPT_CH3_INTR0_EN, VTSS_10G_GPIO_AGGR_INTRPT_CH3_INTR1_EN,
VTSS_10G_GPIO_AGGR_INTRPT_IP1588_0_INTR0_EN, VTSS_10G_GPIO_AGGR_INTRPT_IP1588_0_INTR1_EN,
VTSS_10G_GPIO_AGGR_INTRPT_IP1588_0_INTR2_EN, VTSS_10G_GPIO_AGGR_INTRPT_IP1588_0_INTR3_EN,
VTSS_10G_GPIO_AGGR_INTRPT_IP1588_1_INTR0_EN, VTSS_10G_GPIO_AGGR_INTRPT_IP1588_1_INTR1_EN,
VTSS_10G_GPIO_AGGR_INTRPT_IP1588_1_INTR2_EN, VTSS_10G_GPIO_AGGR_INTRPT_IP1588_1_INTR3_EN,
```

```
VTSS_10G_GPIO_AGGR_INTRPT_LCPLL_0_INTR_EN, VTSS_10G_GPIO_AGGR_INTRPT_LCPLL_1_INTR_EN,
VTSS_10G_GPIO_AGGR_INTRPT_EXP4_INTR_EN, VTSS_10G_GPIO_AGGR_INTRPT_CLK_MUX_INTR_EN,
VTSS_10G_GPIO_AGGR_INTRPT_GPIO_INTR_EN }
```

*GPIO Channel level interrupts.*

- enum `vtss_gpio_10g_input_t` { `VTSS_10G_GPIO_INPUT_NONE`, `VTSS_10G_GPIO_INPUT_LINE_LOPC`,  
`VTSS_10G_GPIO_INPUT_HOST_LOPC` }

*GPIO Channel level interrupts.*

## Functions

- `vtss_rc vtss_phy_10g_mode_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_10g_mode_t` \*const mode)  
*Get the Phy operating mode.*
- `vtss_rc vtss_phy_10g_init` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_10g_init_parm_t` \*const init\_conf)  
*Identify PHY and initialize software accordingly.*
- `vtss_rc vtss_phy_10g_mode_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_10g_mode_t` \*const mode)  
*Identify, Reset and set the operating mode of the PHY.*
- `vtss_rc vtss_phy_10g_ib_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_10g_ib_conf_t` \*const ib\_conf, `BOOL` is\_host)  
*Configure Input buffer .*
- `vtss_rc vtss_phy_10g_ib_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `BOOL` is\_host, `vtss_phy_10g_ib_conf_t` \*const ib\_conf)  
*Get configuration of Input buffer .*
- `vtss_rc vtss_phy_10g_ib_status_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_10g_ib_status_t` \*const ib\_status)  
*Get status of Input buffer .*
- `vtss_rc vtss_phy_10g_apc_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_10g_apc_conf_t` \*const apc\_conf, const `BOOL` is\_host)  
*Configure APC .*
- `vtss_rc vtss_phy_10g_apc_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `BOOL` is\_host, `vtss_phy_10g_apc_conf_t` \*const apc\_conf)  
*Get configuration of APC .*
- `vtss_rc vtss_phy_10g_apc_status_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `BOOL` is\_host, `vtss_phy_10g_apc_status_t` \*const apc\_status)  
*Get status of APC.*
- `vtss_rc vtss_phy_10g_apc_restart` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `BOOL` is\_host)  
*Restart of APC - Debug function only.*
- `vtss_rc vtss_phy_10g_jitter_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_10g_jitter_conf_t` \*const jitter\_conf, `BOOL` is\_host)  
*Configure optimised jitter.*
- `vtss_rc vtss_phy_10g_jitter_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_10g_jitter_conf_t` \*jitter\_conf, `BOOL` is\_host)  
*Gets current Jitter configuration.*
- `vtss_rc vtss_phy_10g_jitter_status_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_10g_jitter_conf_t` \*const jitter\_conf, `BOOL` is\_host)  
*Jitter status.*
- `vtss_rc vtss_phy_10g_sync_clkout_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `BOOL` \*const sync\_clkout)  
*Get the status of recovered clock from PHY. (recommended to use vtss\_phy\_10g\_rxckout\_get instead)*

- `vtss_rc vtss_phy_10g_sync_clkout_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `BOOL` sync\_clkout)
 

*Enable or Disable the recovered clock from PHY. (recommended to use `vtss_phy_10g_rxckout_set` instead)*
- `vtss_rc vtss_phy_10g_xfp_clkout_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `BOOL` \*const xfp\_clkout)
 

*Get the status of RXCLKOUT/TXCLKOUT from PHY. (recommended to use `vtss_phy_10g_txckout_get` instead)*
- `vtss_rc vtss_phy_10g_xfp_clkout_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `BOOL` xfp\_clkout)
 

*Enable or Disable the RXCLKOUT/TXCLKOUT from PHY. (recommended to use `vtss_phy_10g_txckout_set` instead)*
- `vtss_rc vtss_phy_10g_rxckout_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_10g_rxckout_conf_t` \*const rxckout)
 

*Get the rx recovered clock output configuration.*
- `vtss_rc vtss_phy_10g_rxckout_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_10g_rxckout_conf_t` \*const rxckout)
 

*Set the rx recovered clock output configuration.*
- `vtss_rc vtss_phy_10g_txckout_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_10g_txckout_conf_t` \*const txckout)
 

*Get the status of tx recovered clock output configuration.*
- `vtss_rc vtss_phy_10g_txckout_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_10g_txckout_conf_t` \*const txckout)
 

*Set the tx recovered clock output configuration.*
- `vtss_rc vtss_phy_10g_srefclk_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_10g_srefclk_mode_t` \*const srefclk)
 

*Get the configuration of srefclk setting  
Available for PHY family VENICE  
This function should not be used any more, instead use the API function `vtss_phy_10g_mode_get`, see the parameter documentation for that function.*
- `vtss_rc vtss_phy_10g_srefclk_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_10g_srefclk_mode_t` \*const srefclk)
 

*Set the configuration of srefclk setting. Available for PHY family VENICE  
This function should not be used any more, instead use the API function `vtss_phy_10g_mode_set`, see the parameter documentation for that function.*
- `vtss_rc vtss_phy_10g_sckout_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_10g_sckout_conf_t` \*const sckout)
 

*Set the configuration of sckout setting. Available for PHY family MALIBU*
- `vtss_rc vtss_phy_10g_ckout_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_10g_ckout_conf_t` \*const ckout)
 

*Set the configuration of ckout setting. Available for PHY family MALIBU*
- `vtss_rc vtss_phy_10g_line_clk_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_10g_line_clk_conf_t` \*const line\_clk)
 

*Set the configuration of sckout setting. Available for PHY family MALIBU*
- `vtss_rc vtss_phy_10g_host_clk_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_10g_host_clk_conf_t` \*const host\_clk)
 

*Set the configuration of sckout setting. Available for PHY family MALIBU*
- `vtss_rc vtss_phy_10g_line_recvrd_clk_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_10g_line_clk_conf_t` \*const line\_clk)
 

*Set the configuration of line clk recovered setting. Available for PHY family MALIBU*
- `vtss_rc vtss_phy_10g_host_recvrd_clk_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_10g_host_clk_conf_t` \*const host\_clk)
 

*Set the configuration of line clk recovered setting. Available for PHY family MALIBU*

*Set the configuration of host clk recovered setting. Available for PHY family MALIBU*

- `vtss_rc vtss_phy_10g_lane_sync_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_10g_lane_sync_conf_t *const lane_sync)`

*Set the configuration of lane sync setting. Available for PHY family MALIBU*

- `vtss_rc vtss_phy_10g_debug_register_dump (const vtss_inst_t inst, const vtss_debug_printf_t pr, BOOL clear, const vtss_port_no_t port_no)`

*Set the configuration of 10G PHY Dump setting. Available for PHY family Venice & Malibu*

- `vtss_rc vtss_phy_10g_base_kr_train_aneg_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_10g_base_kr_train_aneg_t *const kr_tr_aneg)`

*Get the configuration of 10g\_base\_kr\_tr\_aneg setting. Available for PHY family Malibu,venice-c.*

- `vtss_rc vtss_phy_10g_base_kr_train_aneg_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_10g_base_kr_train_aneg_t *const kr_tr_aneg)`

*Set the configuration of 10g\_base\_kr\_tr\_aneg setting. Available for PHY family Malibu,venice-c.*

- `vtss_rc vtss_phy_10g_base_host_kr_train_aneg_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_10g_base_kr_train_aneg_t *const kr_tr_aneg)`

*Set the configuration of Host 10g\_base\_kr\_tr\_aneg setting. Available for PHY family Malibu.*

- `vtss_rc vtss_phy_10g_base_kr_conf_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_10g_base_kr_conf_t *const kr_conf)`

*Get the configuration of Line 10G output buffer setting. Available for PHY family VENICE-c,Malibu.*

- `vtss_rc vtss_phy_10g_base_kr_conf_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_10g_base_kr_conf_t *const kr_conf)`

*Set the configuration of Line 10G output buffer. Available for PHY family VENICE-c,MALIBU: Note: The parameters cm1,c0, c1 have a range of -32..31. The Output signal from the KR circuit is symmetric, i.e. the voltage output is configured value + 1/2lsb. (ex: -1 => -1/2lsb voltage level, 0 => +1/2lsb voltage level) The parameter ampl is set in steps of 25 mV, therefore the setting is rounded up to a multiplum og 25 mV, i.e. 1101..1125 => 1125 mVppd.*

- `vtss_rc vtss_phy_10g_base_kr_host_conf_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_10g_base_kr_conf_t *const kr_conf)`

*Get the configuration of Host 10G output buffer.*

- `vtss_rc vtss_phy_10g_base_kr_host_conf_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_10g_base_kr_conf_t *const kr_conf)`

*Set the configuration of host 10G output buffer. Note: The parameters cm1,c0, c1 have a range of -32..31. The Output signal from the KR circuit is symmetric, i.e. the voltage output is configured value + 1/2lsb. (ex: -1 => -1/2lsb voltage level, 0 => +1/2lsb voltage level) The parameter ampl is set in steps of 25 mV, therefore the setting is rounded up to a multiplum og 25 mV, i.e. 1101..1125 => 1125 mVppd.*

- `vtss_rc vtss_phy_10g_kr_status_get (const vtss_inst_t inst, const vtss_port_no_t port_no, BOOL direction, vtss_phy_10g_base_kr_status_t *const kr_status)`

*Get status of KR autonegotiation and training. Available for PHY family Malibu,Venice-c.*

- `vtss_rc vtss_phy_10g_ob_status_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_10g_ob_status_t *const ob_status)`

*Get the status of Output Buffer.*

- `vtss_rc vtss_phy_10g_status_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_10g_status_t *const status)`

*Get the link and fault status of the PHY sublayers.*

- `vtss_rc vtss_phy_10g_serd़es_status_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_10g_serd़es_status_t *const status)`

*Get the status of PHY including sub layers.*

- `vtss_rc vtss_phy_10g_reset (const vtss_inst_t inst, const vtss_port_no_t port_no)`

*Reset the phy. Phy is reset to default values.*

- `vtss_rc vtss_phy_10g_clause_37_status_get (const vtss_inst_t inst, vtss_port_no_t port_no, vtss_phy_10g_clause_37_cmn_status_t *const status)`

*Get clause 37 status.*

- `vtss_rc vtss_phy_10g_clause_37_control_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_10g_clause_37_control_t` \*const control)
 

*Get clause 37 control configuration from software.*
- `vtss_rc vtss_phy_10g_clause_37_control_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_10g_clause_37_control_t` \*const control)
 

*Set clause 37 control configuration.*
- `vtss_rc vtss_phy_10g_loopback_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_10g_loopback_t` \*const loopback)
 

*Enable/Disable a phy network or system loopback.*  
*Only one loopback mode can be active at the same time.*
- `vtss_rc vtss_phy_10g_loopback_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_10g_loopback_t` \*const loopback)
 

*Get loopback settings.*
- `vtss_rc vtss_phy_10g_cnt_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_10g_cnt_t` \*const cnt)
 

*Get counters.*
- `vtss_rc vtss_phy_10g_power_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_10g_power_t` \*const power)
 

*Get the power settings.*
- `vtss_rc vtss_phy_10g_power_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_10g_power_t` \*const power)
 

*Set the power settings.*
- `BOOL vtss_phy_10G_is_valid` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no)
 

*Gives a True/False value if the Phy is supported by the API*  
*Only Vitesse phys are supported. `vtss_phy_10g_mode_set()` must be applied.*
- `vtss_rc vtss_phy_10g_failover_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_10g_failover_mode_t` \*const mode)
 

*Set the failover mode.*
- `vtss_rc vtss_phy_10g_failover_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_10g_failover_mode_t` \*const mode)
 

*Get the failover mode.*
- `vtss_rc vtss_phy_10g_auto_failover_set` (const `vtss_inst_t` inst, `vtss_phy_10g_auto_failover_conf_t` \*const mode)
 

*Set the automatic failover mode.*
- `vtss_rc vtss_phy_10g_auto_failover_get` (const `vtss_inst_t` inst, `vtss_phy_10g_auto_failover_conf_t` \*const mode)
 

*Get the Automatic failover mode Configuration.*
- `vtss_rc vtss_phy_10g_vscope_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_10g_vscope_conf_t` \*const conf)
 

*set VSCOPE fast scan configuration*
- `vtss_rc vtss_phy_10g_vscope_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_10g_vscope_conf_t` \*const conf)
 

*get VSCOPE fast scan configuration*
- `vtss_rc vtss_phy_10g_vscope_scan_status_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_10g_vscope_scan_status_t` \*const conf)
 

*set VSCOPE fast scan configuration*
- `vtss_rc vtss_phy_10g_pcs_prbs_gen_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_10g_pcs_prbs_gen_conf_t` \*const conf, const `BOOL` line)
 

*Set PCS-prbs generator Configuration.*
- `vtss_rc vtss_phy_10g_pcs_prbs_gen_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_10g_pcs_prbs_gen_conf_t` \*const conf, const `BOOL` line)
 

*Get PCS-prbs generator Configuration.*

- `vtss_rc vtss_phy_10g_pcs_prbs_mon_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_10g_pcs_prbs_mon_conf_t` \*const conf, const `BOOL` line)
 

*Set PCS-prbs monitor Configuration.*
- `vtss_rc vtss_phy_10g_pcs_prbs_mon_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_10g_pcs_prbs_mon_conf_t` \*const conf, const `BOOL` line)
 

*Get PCS-prbs monitor Configuration.*
- `vtss_rc vtss_phy_10g_pcs_prbs_mon_status_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_10g_pcs_prbs_mon_conf_t` \*const conf, const `BOOL` line)
 

*Get PCS-prbs monitor status.*
- `vtss_rc vtss_phy_10g_prbs_gen_conf` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_10g_prbs_gen_conf_t` \*const conf)
 

*set prbs generator Configuration*
- `vtss_rc vtss_phy_10g_prbs_gen_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_10g_prbs_gen_conf_t` \*const conf, `BOOL` line)
 

*get prbs generator Configuration*
- `vtss_rc vtss_phy_10g_prbs_mon_conf` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_10g_prbs_mon_conf_t` \*const conf)
 

*set prbs generator Configuration*
- `vtss_rc vtss_phy_10g_prbs_mon_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_10g_prbs_mon_conf_t` \*const conf, `BOOL` line)
 

*prbs generator Configuration get*
- `vtss_rc vtss_phy_10g_prbs_mon_status_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_10g_prbs_mon_conf_t` \*const mon\_status, `BOOL` line, `BOOL` reset)
 

*prbs Checker Status get*
- `vtss_rc vtss_phy_10g_pkt_gen_conf` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_10g_pkt_gen_conf_t` \*const conf)
 

*Set Packet generation Configuration.*
- `vtss_rc vtss_phy_10g_pkt_mon_conf` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `BOOL` ts\_rd, `vtss_phy_10g_pkt_mon_conf_t` \*const conf, `vtss_phy_10g_timestamp_val_t` \*const conf\_ts)
 

*Set Packet Monitor Configuration.*
- `vtss_rc vtss_phy_10g_pkt_mon_counters_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_10g_pkt_mon_conf_t` \*const conf)
 

*Set/Get Packet mon Counters.*
- `vtss_rc vtss_phy_10g_id_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_10g_id_t` \*const phy\_id)
 

*Read the Phy Id.*
- `vtss_rc vtss_phy_10g_gpio_mode_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_gpio_10g_no_t` gpio\_no, const `vtss_gpio_10g_gpio_mode_t` \*const mode)
 

*Set GPIO mode. There is only one set of GPIO per PHY chip - not per port.*
- `vtss_rc vtss_phy_10g_gpio_mode_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_gpio_10g_no_t` gpio\_no, `vtss_gpio_10g_gpio_mode_t` \*const mode)
 

*Get GPIO mode.*
- `vtss_rc vtss_phy_10g_gpio_read` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_gpio_10g_no_t` gpio\_no, `BOOL` \*const value)
 

*Read from GPIO input pin.*
- `vtss_rc vtss_phy_10g_gpio_write` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_gpio_10g_no_t` gpio\_no, const `BOOL` value)
 

*Write to GPIO output pin.*
- `vtss_rc vtss_phy_10g_event_enable_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_10g_event_t` ev\_mask, const `BOOL` enable)
 

*Enabling / Disabling of events.*
- `vtss_rc vtss_phy_10g_event_enable_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_10g_event_t` \*const ev\_mask)

- Get Enabling of events.*
- `vtss_rc vtss_phy_10g_extended_event_enable_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_10g_extnd_event_t` \*const ex\_ev\_mask)
- Get Enabling of events.*
- `vtss_rc vtss_phy_10g_event_poll` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_10g_event_t` \*const ev\_mask)
- Polling for active events.*
- `vtss_rc vtss_phy_10g_pcs_status_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_10g_extnd_event_t` \*const ex\_events)
- poll and clear PCS STICKY Register*
- `vtss_rc vtss_phy_10g_extended_event_poll` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_10g_extnd_event_t` \*const ex\_events)
- Polling for active events.*
- `vtss_rc vtss_phy_10g_extended_event_enable_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_10g_extnd_event_t` ex\_ev\_mask, const `BOOL` extnd\_enable)
- Enabling / Disabling of events.*
- `vtss_rc vtss_phy_10g_poll_1sec` (const `vtss_inst_t` inst)
- Function is called once a second.*
- `vtss_rc vtss_phy_10g_edc_fw_status_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_10g_fw_status_t` \*const status)
- Internal microprocessor status.*
- `vtss_rc vtss_phy_10g_fc_buffer_reset` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no)
- debug function for PHY 10G FC buffer reset*
- `vtss_rc vtss_phy_10g_csr_read` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `u32` dev, const `u32` addr, `u32` \*const value)
- CSR register read.*
- `vtss_rc vtss_phy_10g_csr_write` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `u32` dev, const `u32` addr, const `u32` value)
- CSR register write.*
- `vtss_rc vtss_phy_warm_start_10g_failed_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no)
- Function for checking if any issue were seen during warm-start.*
- `vtss_rc vtss_phy_10g_sgmii_mode_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `BOOL` enable)
- Enables Pass through mode in 10G PHY.*
- `vtss_rc vtss_phy_10g_i2c_read` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `u16` addr, `u16` \*value)
- read from i2c device*
- `vtss_rc vtss_phy_10g_i2c_write` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `u16` addr, const `u16` \*value)
- Write to i2c device.*
- `vtss_rc vtss_phy_10g_get_user_data` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, void \*\*user\_data)
- Gets generic pointer in vtss\_state structure.*

### 6.30.1 Detailed Description

#### 10G PHY API

This header file describes 10G PHY control functions

### 6.30.2 Macro Definition Documentation

### 6.30.2.1 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.

### 6.30.2.2 VTSS\_SLEWRATE\_35PS

```
#define VTSS_SLEWRATE_35PS 1
```

Slewrate = 35ps

Definition at line 1187 of file vtss\_phy\_10g\_api.h.

### 6.30.2.3 VTSS\_SLEWRATE\_55PS

```
#define VTSS_SLEWRATE_55PS 2
```

Slewrate = 55ps

Definition at line 1188 of file vtss\_phy\_10g\_api.h.

### 6.30.2.4 VTSS\_SLEWRATE\_70PS

```
#define VTSS_SLEWRATE_70PS 3
```

Slewrate = 70ps

Definition at line 1189 of file vtss\_phy\_10g\_api.h.

### 6.30.2.5 VTSS\_SLEWRATE\_120PS

```
#define VTSS_SLEWRATE_120PS 4
```

Slewrate = 120ps

Definition at line 1190 of file vtss\_phy\_10g\_api.h.

### 6.30.2.6 VTSS\_SLEWRATE\_INVALID

```
#define VTSS_SLEWRATE_INVALID 5
```

Slewrate is invalid

Definition at line 1191 of file vtss\_phy\_10g\_api.h.

### 6.30.2.7 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.

### 6.30.2.8 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.

### 6.30.2.9 PHASE\_POINTS

```
#define PHASE_POINTS 128
```

phase points range from 0-127

Definition at line 1915 of file vtss\_phy\_10g\_api.h.

### 6.30.2.10 AMPLITUDE\_POINTS

```
#define AMPLITUDE_POINTS 64
```

amplitude points range from 0-63

Definition at line 1916 of file vtss\_phy\_10g\_api.h.

### 6.30.2.11 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.

### 6.30.2.12 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.

### 6.30.2.13 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.

### 6.30.2.14 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.

### 6.30.2.15 VTSS\_10G\_PHY\_GPIO\_MAX

```
#define VTSS_10G_PHY_GPIO_MAX 12
```

Max value of gpio\_no parameter

Definition at line 2496 of file vtss\_phy\_10g\_api.h.

### 6.30.2.16 VTSS\_10G\_PHY\_GPIO\_MAL\_MAX

```
#define VTSS_10G_PHY_GPIO_MAL_MAX 40
```

Max value of gpio\_no parameter, Malibu

Definition at line 2498 of file vtss\_phy\_10g\_api.h.

### 6.30.2.17 VTSS\_PHY\_10G\_LINK\_LOS\_EV

```
#define VTSS_PHY_10G_LINK_LOS_EV 0x00000001
```

Event source identification mask values.

PHY Link Los interrupt - only on 8486

Definition at line 2563 of file vtss\_phy\_10g\_api.h.

### 6.30.2.18 VTSS\_PHY\_10G\_RX\_LOL\_EV [1/2]

```
#define VTSS_PHY_10G_RX_LOL_EV 0x00000002
```

PHY RXLOL interrupt - only on 8488

PHY RX LOL interrupt - 8256 specific

Definition at line 2603 of file vtss\_phy\_10g\_api.h.

### 6.30.2.19 VTSS\_PHY\_10G\_TX\_LOL\_EV [1/2]

```
#define VTSS_PHY_10G_TX_LOL_EV 0x00000004
```

PHY TXLOL interrupt - only on 8488

PHY TX LOL interrupt - 8256 specific

Definition at line 2604 of file vtss\_phy\_10g\_api.h.

### 6.30.2.20 VTSS\_PHY\_10G\_LOPC\_EV

```
#define VTSS_PHY_10G_LOPC_EV 0x00000008
```

PHY LOPC interrupt - only on 8488

Definition at line 2566 of file vtss\_phy\_10g\_api.h.

**6.30.2.21 VTSS\_PHY\_10G\_HIGH\_BER\_EV**

```
#define VTSS_PHY_10G_HIGH_BER_EV 0x00000010
```

PHY HIGH\_BER interrupt - only on 8488

Definition at line 2567 of file vtss\_phy\_10g\_api.h.

**6.30.2.22 VTSS\_PHY\_10G\_MODULE\_STAT\_EV**

```
#define VTSS_PHY_10G_MODULE_STAT_EV 0x00000020
```

PHY MODULE\_STAT interrupt - only on 8488

Definition at line 2568 of file vtss\_phy\_10g\_api.h.

**6.30.2.23 VTSS\_PHY\_10G\_PCS\_RECEIVE\_FAULT\_EV**

```
#define VTSS_PHY_10G_PCS_RECEIVE_FAULT_EV 0x00000040
```

PHY PCS\_RECEIVE\_FAULT interrupt - only on 8488

Definition at line 2569 of file vtss\_phy\_10g\_api.h.

**6.30.2.24 VTSS\_PHY\_EWIS\_SEF\_EV**

```
#define VTSS_PHY_EWIS_SEF_EV 0x00000080
```

SEF has changed state - only for 8488

Definition at line 2571 of file vtss\_phy\_10g\_api.h.

**6.30.2.25 VTSS\_PHY\_EWIS\_FPLM\_EV**

```
#define VTSS_PHY_EWIS_FPLM_EV 0x00000100
```

far-end (PLM-P) / (LCDP) - only for 8488

Definition at line 2572 of file vtss\_phy\_10g\_api.h.

### 6.30.2.26 VTSS\_PHY\_EWIS\_FAIS\_EV

```
#define VTSS_PHY_EWIS_FAIS_EV 0x00000200
```

far-end (AIS-P) / (LOP) - only for 8488

Definition at line 2573 of file vtss\_phy\_10g\_api.h.

### 6.30.2.27 VTSS\_PHY\_EWIS\_LOF\_EV

```
#define VTSS_PHY_EWIS_LOF_EV 0x00000400
```

Loss of Frame (LOF) - only for 8488

Definition at line 2574 of file vtss\_phy\_10g\_api.h.

### 6.30.2.28 VTSS\_PHY\_EWIS\_RDIL\_EV

```
#define VTSS_PHY_EWIS_RDIL_EV 0x00000800
```

Line Remote Defect Indication (RDI-L) - only for 8488

Definition at line 2575 of file vtss\_phy\_10g\_api.h.

### 6.30.2.29 VTSS\_PHY\_EWIS\_AISL\_EV

```
#define VTSS_PHY_EWIS_AISL_EV 0x00001000
```

Line Alarm Indication Signal (AIS-L) - only for 8488

Definition at line 2576 of file vtss\_phy\_10g\_api.h.

### 6.30.2.30 VTSS\_PHY\_EWIS\_LCDP\_EV

```
#define VTSS_PHY_EWIS_LCDP_EV 0x00002000
```

Loss of Code-group Delineation (LCD-P) - only for 8488

Definition at line 2577 of file vtss\_phy\_10g\_api.h.

### 6.30.2.31 VTSS\_PHY\_EWIS\_PLMP\_EV

```
#define VTSS_PHY_EWIS_PLMP_EV 0x00004000
```

Path Label Mismatch (PLMP) - only for 8488

Definition at line 2578 of file vtss\_phy\_10g\_api.h.

### 6.30.2.32 VTSS\_PHY\_EWIS\_AISP\_EV

```
#define VTSS_PHY_EWIS_AISP_EV 0x00008000
```

Path Alarm Indication Signal (AIS-P) - only for 8488

Definition at line 2579 of file vtss\_phy\_10g\_api.h.

### 6.30.2.33 VTSS\_PHY\_EWIS\_LOPP\_EV

```
#define VTSS_PHY_EWIS_LOPP_EV 0x00010000
```

Path Loss of Pointer (LOP-P) - only for 8488

Definition at line 2580 of file vtss\_phy\_10g\_api.h.

### 6.30.2.34 VTSS\_PHY\_EWIS\_UNEQP\_EV

```
#define VTSS_PHY_EWIS_UNEQP_EV 0x00020000
```

Unequiped Path (UNEQ-P) - only for 8488

Definition at line 2581 of file vtss\_phy\_10g\_api.h.

### 6.30.2.35 VTSS\_PHY\_EWIS\_FEUNEQP\_EV

```
#define VTSS_PHY_EWIS_FEUNEQP_EV 0x00040000
```

Far-end Unequiped Path (UNEQ-P) - only for 8488

Definition at line 2582 of file vtss\_phy\_10g\_api.h.

### 6.30.2.36 VTSS\_PHY\_EWIS\_FERDIP\_EV

```
#define VTSS_PHY_EWIS_FERDIP_EV 0x00080000
```

Far-end Path Remote Defect Identifier (RDI-P) - only for 8488

Definition at line 2583 of file vtss\_phy\_10g\_api.h.

### 6.30.2.37 VTSS\_PHY\_EWIS\_REIL\_EV

```
#define VTSS_PHY_EWIS_REIL_EV 0x00100000
```

Line Remote Error Indication (REI-L) - only for 8488

Definition at line 2584 of file vtss\_phy\_10g\_api.h.

### 6.30.2.38 VTSS\_PHY\_EWIS\_REIP\_EV

```
#define VTSS_PHY_EWIS_REIP_EV 0x00200000
```

Path Remote Error Indication (REI-P) - only for 8488

Definition at line 2585 of file vtss\_phy\_10g\_api.h.

### 6.30.2.39 VTSS\_PHY\_EWIS\_B1\_NZ\_EV

```
#define VTSS_PHY_EWIS_B1_NZ_EV 0x00400000
```

PMTICK B1 BIP (B1\_ERR\_CNT) not zero - only for 8488

Definition at line 2586 of file vtss\_phy\_10g\_api.h.

### 6.30.2.40 VTSS\_PHY\_EWIS\_B2\_NZ\_EV

```
#define VTSS_PHY_EWIS_B2_NZ_EV 0x00800000
```

PMTICK B2 BIP (B1\_ERR\_CNT) not zero - only for 8488

Definition at line 2587 of file vtss\_phy\_10g\_api.h.

### 6.30.2.41 VTSS\_PHY\_EWIS\_B3\_NZ\_EV

```
#define VTSS_PHY_EWIS_B3_NZ_EV 0x01000000
```

PMTICK B3 BIP (B1\_ERR\_CNT) not zero - only for 8488

Definition at line 2588 of file vtss\_phy\_10g\_api.h.

### 6.30.2.42 VTSS\_PHY\_EWIS\_REIL\_NZ\_EV

```
#define VTSS_PHY_EWIS_REIL_NZ_EV 0x02000000
```

PMTICK REI-L (REIL\_ERR\_CNT) not zero - only for 8488

Definition at line 2589 of file vtss\_phy\_10g\_api.h.

### 6.30.2.43 VTSS\_PHY\_EWIS\_REIP\_NZ\_EV

```
#define VTSS_PHY_EWIS_REIP_NZ_EV 0x04000000
```

PMTICK REI-P (REIP\_ERR\_CNT) not zero - only for 8488

Definition at line 2590 of file vtss\_phy\_10g\_api.h.

### 6.30.2.44 VTSS\_PHY\_EWIS\_B1\_THRESH\_EV

```
#define VTSS_PHY_EWIS_B1_THRESH_EV 0x08000000
```

B1\_THRESH\_ERR - only for 8488

Definition at line 2591 of file vtss\_phy\_10g\_api.h.

### 6.30.2.45 VTSS\_PHY\_EWIS\_B2\_THRESH\_EV

```
#define VTSS_PHY_EWIS_B2_THRESH_EV 0x10000000
```

B2\_THRESH\_ERR - only for 8488

Definition at line 2592 of file vtss\_phy\_10g\_api.h.

### 6.30.2.46 VTSS\_PHY\_EWIS\_B3\_THRESH\_EV

```
#define VTSS_PHY_EWIS_B3_THRESH_EV 0x20000000
```

B3\_THRESH\_ERR - only for 8488

Definition at line 2593 of file vtss\_phy\_10g\_api.h.

### 6.30.2.47 VTSS\_PHY\_EWIS\_REIL\_THRESH\_EV

```
#define VTSS_PHY_EWIS_REIL_THRESH_EV 0x40000000
```

REIL\_THRESH\_ERR - only for 8488

Definition at line 2594 of file vtss\_phy\_10g\_api.h.

### 6.30.2.48 VTSS\_PHY\_EWIS\_REIP\_THRESH\_EV

```
#define VTSS_PHY_EWIS_REIP_THRESH_EV 0x80000000
```

REIP\_THRESH\_ERR - only for 8488

Definition at line 2595 of file vtss\_phy\_10g\_api.h.

### 6.30.2.49 VTSS\_PHY\_10G\_RX\_LOS\_EV

```
#define VTSS_PHY_10G_RX_LOS_EV 0x00000001
```

PHY RX LOS interrupt - 8256 specific

Definition at line 2602 of file vtss\_phy\_10g\_api.h.

### 6.30.2.50 VTSS\_PHY\_10G\_RX\_LOL\_EV [2/2]

```
#define VTSS_PHY_10G_RX_LOL_EV 0x00000002
```

PHY RX LOL interrupt - only on 8488

PHY RX LOL interrupt - 8256 specific

Definition at line 2603 of file vtss\_phy\_10g\_api.h.

**6.30.2.51 VTSS\_PHY\_10G\_TX\_LOL\_EV [2/2]**

```
#define VTSS_PHY_10G_TX_LOL_EV 0x00000004
```

PHY TXLOL interrupt - only on 8488

PHY TX LOL interrupt - 8256 specific

Definition at line 2604 of file vtss\_phy\_10g\_api.h.

**6.30.2.52 VTSS\_PHY\_10G\_RX\_CHAR\_DEC\_CNT\_THRESH\_EV**

```
#define VTSS_PHY_10G_RX_CHAR_DEC_CNT_THRESH_EV 0x00000010
```

PHY RX character decode error - 8256 specific

Definition at line 2606 of file vtss\_phy\_10g\_api.h.

**6.30.2.53 VTSS\_PHY\_10G\_TX\_CHAR\_ENC\_CNT\_THRESH\_EV**

```
#define VTSS_PHY_10G_TX_CHAR_ENC_CNT_THRESH_EV 0x00000020
```

PHY TX character encode error count - 8256 specific

Definition at line 2607 of file vtss\_phy\_10g\_api.h.

**6.30.2.54 VTSS\_PHY\_10G\_RX\_BLK\_DEC\_CNT\_THRESH\_EV**

```
#define VTSS_PHY_10G_RX_BLK_DEC_CNT_THRESH_EV 0x00000040
```

PHY RX block decode error count - 8256 specific

Definition at line 2608 of file vtss\_phy\_10g\_api.h.

**6.30.2.55 VTSS\_PHY\_10G\_TX\_BLK\_ENC\_CNT\_THRESH\_EV**

```
#define VTSS_PHY_10G_TX_BLK_ENC_CNT_THRESH_EV 0x00000080
```

PHY TX block encode error count- 8256 specific

Definition at line 2609 of file vtss\_phy\_10g\_api.h.

**6.30.2.56 VTSS\_PHY\_10G\_RX\_SEQ\_CNT\_THRESH\_EV**

```
#define VTSS_PHY_10G_RX_SEQ_CNT_THRESH_EV 0x00000100
```

PHY RX sequencing error count - 8256 specific

Definition at line 2610 of file vtss\_phy\_10g\_api.h.

**6.30.2.57 VTSS\_PHY\_10G\_TX\_SEQ\_CNT\_THRESH\_EV**

```
#define VTSS_PHY_10G_TX_SEQ_CNT_THRESH_EV 0x00000200
```

PHY TX sequencing error count - 8256 specific

Definition at line 2611 of file vtss\_phy\_10g\_api.h.

**6.30.2.58 VTSS\_PHY\_10G\_FEC\_UNFIXED\_CNT\_THRESH\_EV**

```
#define VTSS_PHY_10G_FEC_UNFIXED_CNT_THRESH_EV 0x00000400
```

PHY KR-FEC uncorrectable block count interrupt - 8256 specific

Definition at line 2612 of file vtss\_phy\_10g\_api.h.

**6.30.2.59 VTSS\_PHY\_10G\_FEC\_FIXED\_CNT\_THRESH\_EV**

```
#define VTSS_PHY_10G_FEC_FIXED_CNT_THRESH_EV 0x00000800
```

PHY KR-FEC corrected threshold - 8256 specific

Definition at line 2613 of file vtss\_phy\_10g\_api.h.

**6.30.2.60 VTSS\_PHY\_10G\_HIGHBER\_EV**

```
#define VTSS_PHY_10G_HIGHBER_EV 0x00001000
```

PHY high bit Error - 8256 specific

Definition at line 2614 of file vtss\_phy\_10g\_api.h.

**6.30.2.61 VTSS\_PHY\_10G\_RX\_LINK\_STAT\_EV [1/2]**

```
#define VTSS_PHY_10G_RX_LINK_STAT_EV 0x00002000
```

PHY Link status up/down interrupt - 8256 specific

Definition at line 2616 of file vtss\_phy\_10g\_api.h.

**6.30.2.62 VTSS\_PHY\_10G\_RX\_LINK\_STAT\_EV [2/2]**

```
#define VTSS_PHY_10G_RX_LINK_STAT_EV 0x00002000
```

PHY Link status up/down interrupt - 8256 specific

Definition at line 2616 of file vtss\_phy\_10g\_api.h.

**6.30.2.63 VTSS\_PHY\_10G\_GPIO\_INT\_AGG0\_EV**

```
#define VTSS_PHY_10G_GPIO_INT_AGG0_EV 0x00004000
```

PHY GPIO interrupt on Aggregator0 - 8256 specific

Definition at line 2617 of file vtss\_phy\_10g\_api.h.

**6.30.2.64 VTSS\_PHY\_10G\_GPIO\_INT\_AGG1\_EV**

```
#define VTSS_PHY_10G_GPIO_INT_AGG1_EV 0x00008000
```

PHY GPIO interrupt on Aggregator1 - 8256 specific

Definition at line 2618 of file vtss\_phy\_10g\_api.h.

**6.30.2.65 VTSS\_PHY\_10G\_GPIO\_INT\_AGG2\_EV**

```
#define VTSS_PHY_10G_GPIO_INT_AGG2_EV 0x00010000
```

PHY GPIO interrupt on Aggregator2 - 8256 specific

Definition at line 2619 of file vtss\_phy\_10g\_api.h.

**6.30.2.66 VTSS\_PHY\_10G\_GPIO\_INT\_AGG3\_EV**

```
#define VTSS_PHY_10G_GPIO_INT_AGG3_EV 0x00020000
```

PHY GPIO interrupt on Aggregator3 - 8256 specific

Definition at line 2620 of file vtss\_phy\_10g\_api.h.

**6.30.2.67 VTSS\_PHY\_1G\_LINE\_AUTONEG\_RESTART\_EV**

```
#define VTSS_PHY_1G_LINE_AUTONEG_RESTART_EV 0x00040000
```

PHY 1G Line side Autoneg restart event

Definition at line 2621 of file vtss\_phy\_10g\_api.h.

**6.30.2.68 VTSS\_PHY\_1G\_HOST\_AUTONEG\_RESTART\_EV**

```
#define VTSS_PHY_1G_HOST_AUTONEG_RESTART_EV 0x00080000
```

PHY 1G Host side Autoneg restart event - 8256 specific

Definition at line 2622 of file vtss\_phy\_10g\_api.h.

**6.30.2.69 VTSS\_PHY\_10G\_LINE\_MAC\_LOCAL\_FAULT\_EV**

```
#define VTSS_PHY_10G_LINE_MAC_LOCAL_FAULT_EV 0x00100000
```

PHY 10G LINE MAC local fault event

Definition at line 2625 of file vtss\_phy\_10g\_api.h.

**6.30.2.70 VTSS\_PHY\_10G\_HOST\_MAC\_LOCAL\_FAULT\_EV**

```
#define VTSS_PHY_10G_HOST_MAC_LOCAL_FAULT_EV 0x00400000
```

PHY 10G HOST MAC local fault event

Definition at line 2626 of file vtss\_phy\_10g\_api.h.

### 6.30.2.71 VTSS\_PHY\_10G\_LINE\_MAC\_REMOTE\_FAULT\_EV

```
#define VTSS_PHY_10G_LINE_MAC_REMOTE_FAULT_EV 0x00800000
```

PHY 10G LINE MAC remote fault event

Definition at line 2627 of file vtss\_phy\_10g\_api.h.

### 6.30.2.72 VTSS\_PHY\_10G\_HOST\_MAC\_REMOTE\_FAULT\_EV

```
#define VTSS_PHY_10G_HOST_MAC_REMOTE_FAULT_EV 0x01000000
```

PHY 10G HOST MAC remote fault event

Definition at line 2628 of file vtss\_phy\_10g\_api.h.

## 6.30.3 Typedef Documentation

### 6.30.3.1 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.

### 6.30.3.2 ckout\_sel\_t

```
typedef enum ckout_sel_ ckout_sel_t
```

10G Phy CKOUTs Enum

Malibu Only

### 6.30.3.3 vtss\_32\_cntr\_t

```
typedef u32 vtss_32_cntr_t
```

32-bit counter

Definition at line 2175 of file vtss\_phy\_10g\_api.h.

#### 6.30.3.4 vtss\_phy\_10g\_event\_t

```
typedef u32 vtss_phy_10g_event_t
```

The type definition to contain the above defined event mask

Definition at line 2597 of file vtss\_phy\_10g\_api.h.

#### 6.30.3.5 vtss\_phy\_10g\_extnd\_event\_t

```
typedef u32 vtss_phy_10g_extnd_event_t
```

The type definition to contain the above defined extended event mask

Definition at line 2631 of file vtss\_phy\_10g\_api.h.

### 6.30.4 Enumeration Type Documentation

#### 6.30.4.1 oper\_mode\_t

```
enum oper_mode_t
```

10G Phy operating mode enum type

##### Enumerator

VTSS_PHY_LAN_MODE	LAN mode: Single clock (XREFCK=156,25 MHz), no recovered clock output
VTSS_PHY_WAN_MODE	WAN mode: 848X: Dual clock (XREFCK=156,25 MHz, WREFCK=155,52 MHz), no recovered clock output Venice: Single clock (XREFCK), no recovered clock output
VTSS_PHY_1G_MODE	8488: 1G pass-through mode Venice: 1G mode, Single clock (XREFCK=156,25 MHz), no recovered clock output For 1G operation, customer should select VTSS_MEDIA_TYPE_SR for all media applications and specify the operation is in 1G mode
VTSS_PHY_LAN_SYNCE_MODE	LAN SyncE: if hl_clk_synth == 1: 8488: Single clock (XREFCK=156,25 MHz), recovered clock output enabled Venice: Single clock (XREFCK=156,25 MHz), recovered clock output enabled if hl_clk_synth == 0: 8488: Dual clock (XREFCK=156,25 MHz, SREFCK=156,25 MHz), recovered clock output enabled Venice: Dual clock (XREFCK=156,25 MHz, SREFCK=156,25 MHz), recovered clock output enabled

## Enumerator

VTSS_PHY_WAN_SYNCE_MODE	<p>WAN SyncE:          if hl_clk_synth == 1:          8488: Single clock (WREFCK=155,52 MHz or 622,08 MHz), recovered clock output enabled          Venice: Single clock (XREFCK=156,25 MHz), recovered clock output enabled          if hl_clk_synth == 0:          8488: Dual clock (WREFCK=155,52 MHz or 622,08 MHz, SREFCK=155,52 MHz), recovered clock output enabled          Venice: Dual clock (XREFCK=156,25 MHz, SREFCK=155,52 MHz), recovered clock output enabled</p>
VTSS_PHY_LAN_MIXED_SYNCE_MODE	8488: Channels are in different modes, channel being configured is in LAN Venice: Same as VTSS_PHY_LAN_SYNCE_MODE
VTSS_PHY_WAN_MIXED_SYNCE_MODE	8488: Channels are in different modes, channel being configured is in WAN Venice: Same as VTSS_PHY_WAN_SYNCE_MODE
VTSS_PHY_REPEATERT_MODE	Malibu: Repeater mode,better jitter performance

Definition at line 45 of file vtss\_phy\_10g\_api.h.

## 6.30.4.2 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.

## 6.30.4.3 vtss\_phy\_interface\_mode

enum [vtss\\_phy\\_interface\\_mode](#)

Phy Interface modes.

## Enumerator

VTSS_PHY_XAUI_XFI	XAUI <-> XFI - Interface mode.
VTSS_PHY_XGMII_XFI	XGMII <-> XFI - Interface mode. Only for VSC8486

**Enumerator**

<code>VTSS_PHY_RXAUI_XFI</code>	RXAUI <-> XFI - Interface mode. Only for Venice
<code>VTSS_PHY_SGMII_LANE_0_XFI</code>	SGMII <-> XFI - LANE 0. Only for Venice
<code>VTSS_PHY_SGMII_LANE_3_XFI</code>	SGMII <-> XFI - LANE 3. Only for Venice
<code>VTSS_PHY_SFI_XFI</code>	SFI <-> XFI - Interface mode. Only for Malibu

Definition at line 94 of file vtss\_phy\_10g\_api.h.

**6.30.4.4 vtss\_recvrd\_t**

```
enum vtss_recvrd_t
```

Modes for recovered clock.

**Enumerator**

<code>VTSS_RECVRD_RXCLKOUT</code>	RXCLKOUT is used for recovered clock
<code>VTSS_RECVRD_TXCLKOUT</code>	TXCLKOUT is used for recovered clock

Definition at line 104 of file vtss\_phy\_10g\_api.h.

**6.30.4.5 vtss\_recvrdclk\_cdr\_div\_t**

```
enum vtss_recvrdclk_cdr_div_t
```

Modes for recovered clock divisor.

**Enumerator**

<code>VTSS_RECVRDCLK_CDR_DIV_64</code>	recovered clock is /64
<code>VTSS_RECVRDCLK_CDR_DIV_66</code>	recovered clock is /66

Definition at line 110 of file vtss\_phy\_10g\_api.h.

**6.30.4.6 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.

**6.30.4.7 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.

**6.30.4.8 apc\_ib\_regulator\_t**

```
enum apc_ib_regulator_t
```

APC Rx regulator mode.

**Enumerator**

VTSS_AP_C_IB_SFP_PLUS_ZR	SFP+ ZR module.
VTSS_AP_C_IB_BACKPLANE	Backplane application.

Definition at line 130 of file vtss\_phy\_10g\_api.h.

**6.30.4.9 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.

**6.30.4.10 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.

**6.30.4.11 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.

#### 6.30.4.12 vtss\_phy\_10g\_media\_t

enum [vtss\\_phy\\_10g\\_media\\_t](#)

10G Phy Media type

Enumerator

VTSS_MEDIA_TYPE_SR	SR,10GBASE-SR
VTSS_MEDIA_TYPE_SR2	SR,10GBASE-SR
VTSS_MEDIA_TYPE_DAC	DAC,Direct attach cable
VTSS_MEDIA_TYPE_ZR	ZR,10GBASE-ZR
VTSS_MEDIA_TYPE_KR	KR,10GBASE-KR
VTSS_MEDIA_TYPE_SR_SC	SR,10GBASE-SR with smart control
VTSS_MEDIA_TYPE_SR2_SC	SR,10GBASE-SR with smart control
VTSS_MEDIA_TYPE_DAC_SC	DAC,Direct attach cable with smart control
VTSS_MEDIA_TYPE_ZR_SC	ZR,10GBASE-ZR with smart control
VTSS_MEDIA_TYPE_ZR2_SC	ZR,10GBASE-ZR with smart control with ld_lev_ini:40
VTSS_MEDIA_TYPE_KR_SC	KR,10GBASE-KR with smart control
VTSS_MEDIA_TYPE_NONE	None

Definition at line 170 of file vtss\_phy\_10g\_api.h.

#### 6.30.4.13 vtss\_phy\_6g\_link\_partner\_distance\_t

enum [vtss\\_phy\\_6g\\_link\\_partner\\_distance\\_t](#)

6G serdes link partner distance selection

Enumerator

VTSS_6G_LINK_SHORT_RANGE	distance between 6G macro and serdes macro of link partner is less (direct connection)
VTSS_6G_LINK_LONG_RANGE	distance between 6G macro and serdes macro of link parter is more (connection via backplanes)

Definition at line 186 of file vtss\_phy\_10g\_api.h.

#### 6.30.4.14 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.

## 6.30.4.15 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.

## 6.30.4.16 vtss\_recvrd\_clkout\_t

```
enum vtss_recvrd_clkout_t
```

Modes for (rx/tx) recovered clock output.

## Enumerator

VTSS_RECVRD_CLKOUT_DISABLE	recovered clock output is disabled
VTSS_RECVRD_CLKOUT_LINE_SIDE_RX_CLOCK	recovered clock output is derived from Lineside Rx clock
VTSS_RECVRD_CLKOUT_LINE_SIDE_TX_CLOCK	recovered clock output is derived from Lineside Tx clock

Definition at line 699 of file vtss\_phy\_10g\_api.h.

#### 6.30.4.17 vtss\_phy\_10g\_srefclk\_freq\_t

enum `vtss_phy_10g_srefclk_freq_t`

10G Phy sref clock input frequency

Enumerator

VTSS_PHY_10G_SREFCLK_156_25	156,25 MHz
VTSS_PHY_10G_SREFCLK_125_00	125,00 MHz
VTSS_PHY_10G_SREFCLK_155_52	155,52 MHz
VTSS_PHY_10G_SREFCLK_INVALID	Other values are not allowed

Definition at line 779 of file vtss\_phy\_10g\_api.h.

#### 6.30.4.18 vtss\_phy\_10g\_ckout\_freq\_t

enum `vtss_phy_10g_ckout_freq_t`

10G Phy clock frequency

Malibu only

Enumerator

VTSS_PHY_10G_CLK_FULL_RATE	LAN:332.25 MHz, WAN:311.04MHz, 1G:125MHz
VTSS_PHY_10G_CLK_DIVIDE_BY_2	LAN:161.12 MHz, WAN:155.52MHz, 1G:62.5MHz
VTSS_PHY_10G_CLK_INVALID	Other values are not allowed

Definition at line 831 of file vtss\_phy\_10g\_api.h.

#### 6.30.4.19 vtss\_ckout\_data\_sel\_t

enum `vtss_ckout_data_sel_t`

Modes for recovered clock output.

Applicable to Malibu only

Enumerator

VTSS_CKOUT_LINE0_TX_CLOCK	Line0 Transmit clock
VTSS_CKOUT_LINE1_TX_CLOCK	Line1 Transmit clock
VTSS_CKOUT_LINE2_TX_CLOCK	Line2 Transmit clock
VTSS_CKOUT_LINE3_TX_CLOCK	Line3 Transmit clock

## Enumerator

VTSS_CKOUT_HOST0_TX_CLOCK	Host0 Transmit clock
VTSS_CKOUT_HOST1_TX_CLOCK	Host1 Transmit clock
VTSS_CKOUT_HOST2_TX_CLOCK	Host2 Transmit clock
VTSS_CKOUT_HOST3_TX_CLOCK	Host3 Transmit clock
VTSS_CKOUT_LINE0_RECVRD_CLOCK	Line0 Recovered clock
VTSS_CKOUT_LINE1_RECVRD_CLOCK	Line1 Recovered clock
VTSS_CKOUT_LINE2_RECVRD_CLOCK	Line2 Recovered clock
VTSS_CKOUT_LINE3_RECVRD_CLOCK	Line3 Recovered clock
VTSS_CKOUT_HOST0_RECVRD_CLOCK	Host0 Recovered clock
VTSS_CKOUT_HOST1_RECVRD_CLOCK	Host1 Recovered clock
VTSS_CKOUT_HOST2_RECVRD_CLOCK	Host2 Recovered clock
VTSS_CKOUT_HOST3_RECVRD_CLOCK	Host3 Recovered clock
VTSS_CKOUT_HOST_PLL_CLOCK	Host PLL clock
VTSS_CKOUT_LINE_PLL_CLOCK	Line PLL clock
VTSS_CKOUT_CSR_CLOCK	CSR clock
VTSS_CKOUT_LTC_CLOCK	LTC clock
VTSS_CKOUT_DF2F_CLOCK	Df2f clock
VTSS_CKOUT_F2DF_CLOCK	F2df clock
VTSS_CKOUT_DEBUG1	Debug1
VTSS_CKOUT_DEBUG2	Debug2
VTSS_CKOUT_OSCILLATOR_OUTPUT	Oscillator output

Definition at line 841 of file vtss\_phy\_10g\_api.h.

## 6.30.4.20 vtss\_phy\_10g\_squelch\_src\_t

enum [vtss\\_phy\\_10g\\_squelch\\_src\\_t](#)

squelch control source

Applicable to Malibu only

## Enumerator

VTSS_CKOUT_SQUELCH_SRC_GPIO0	GPIO0 as source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_GPIO1	GPIO1 as source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_GPIO2	GPIO2 as source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_GPIO3	GPIO3 as source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_GPIO4	GPIO4 as source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_GPIO5	GPIO5 as source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_GPIO6	GPIO6 as source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_GPIO7	GPIO7 as source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_LINK_LINE0	Link status from Line0 source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_LINK_LINE1	Link status from Line1 source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_LINK_LINE2	Link status from Line2 source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_LINK_LINE3	Link status from Line3 source of auto squelch

## Enumerator

VTSS_CKOUT_SQUELCH_SRC_LINK_HOST0	Link status from Host0 source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_LINK_HOST1	Link status from Host1 source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_LINK_HOST2	Link status from Host2 source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_LINK_HOST3	Link status from Host3 source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_LOS_LINE0	Serdes LOS from Line0 as source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_LOS_LINE1	Serdes LOS from Line1 as source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_LOS_LINE2	Serdes LOS from Line2 as source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_LOS_LINE3	Serdes LOS from Line3 as source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_LOS_HOST0	Serdes LOS from Host0 as source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_LOS_HOST1	Serdes LOS from Host1 as source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_LOS_HOST2	Serdes LOS from Host2 as source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_LOS_HOST3	Serdes LOS from Host3 as source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_LINK_LINE0_KR	Link status from Line0 KR source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_LINK_LINE1_KR	Link status from Line1 KR source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_LINK_LINE2_KR	Link status from Line2 KR source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_LINK_LINE3_KR	Link status from Line3 KR source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_LINK_HOST0_KR	Link status from Host0 KR source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_LINK_HOST1_KR	Link status from Host1 KR source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_LINK_HOST2_KR	Link status from Host2 KR source of auto squelch
VTSS_CKOUT_SQUELCH_SRC_LINK_HOST3_KR	Link status from Host3 KR source of auto squelch
VTSS_CKOUT_NO_SQUELCH	No squelch(32-63)

Definition at line 873 of file vtss\_phy\_10g\_api.h.

## 6.30.4.21 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.

#### 6.30.4.22 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

<code>VTSS_PHY_10G_USE_LINE0_RECVRD_CLOCK</code>	All lines using Line0 Recovered clock
<code>VTSS_PHY_10G_USE_LINE1_RECVRD_CLOCK</code>	All lines using Line1 Recovered clock
<code>VTSS_PHY_10G_USE_LINE2_RECVRD_CLOCK</code>	All lines using Line2 Recovered clock
<code>VTSS_PHY_10G_USE_LINE3_RECVRD_CLOCK</code>	All lines using Line3 Recovered clock
<code>VTSS_PHY_10G_USE_HOST0_RECVRD_CLOCK</code>	All lines using Host0 Recovered clock
<code>VTSS_PHY_10G_USE_HOST1_RECVRD_CLOCK</code>	All lines using Host1 Recovered clock
<code>VTSS_PHY_10G_USE_HOST2_RECVRD_CLOCK</code>	All lines using Host2 Recovered clock
<code>VTSS_PHY_10G_USE_HOST3_RECVRD_CLOCK</code>	All lines using Host3 Recovered clock
<code>VTSS_PHY_10G_USE_SREFCLK_CLOCK</code>	All lines using SREFCLK
<code>VTSS_PHY_10G_USE_DEFAULT_RECVRD_CLK↔OCK</code>	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.

#### 6.30.4.23 ckout\_sel\_

enum `ckout_sel_`

10G Phy CKOUTs Enum

Malibu Only

Definition at line 951 of file vtss\_phy\_10g\_api.h.

#### 6.30.4.24 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.

## 6.30.4.25 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.

## 6.30.4.26 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.

## 6.30.4.27 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.

#### 6.30.4.28 vtss\_lb\_type\_t

enum [vtss\\_lb\\_type\\_t](#)

10G loopback types

Enumerator

VTSS_LB_NONE	No looback
VTSS_LB_SYSTEM_XS_SHALLOW	System Loopback B, XAUI -> XS -> XAUI 4x800E.13, Venice: H2
VTSS_LB_SYSTEM_XS_DEEP	System Loopback C, XAUI -> XS -> XAUI 4x800F.2, Venice: N.A.
VTSS_LB_SYSTEM_PCS_SHALLOW	System Loopback E, XAUI -> PCS FIFO -> XAUI 3x8005.2, Venice: N.A.
VTSS_LB_SYSTEM_PCS_DEEP	System Loopback G, XAUI -> PCS -> XAUI 3x0000.14, Venice: H3
VTSS_LB_SYSTEM_PMA	System Loopback J, XAUI -> PMA -> XAUI 1x0000.0, Venice: H4
VTSS_LB_NETWORK_XS_SHALLOW	Network Loopback D, XFI -> XS -> XFI 4x800F.1, Venice: N.A.
VTSS_LB_NETWORK_XS_DEEP	Network Loopback A, XFI -> XS -> XFI 4x0000.1 4x800E.13=0, Venice: L1
VTSS_LB_NETWORK_PCS	Network Loopback F, XFI -> PCS -> XFI 3x8005.3, Venice: L2
VTSS_LB_NETWORK_WIS	Network Loopback H, XFI -> WIS -> XFI 2xE600.0, Venice: N.A.
VTSS_LB_NETWORK_PMA	Network Loopback K, XFI -> PMA -> XFI 1x8000.8, Venice: L3
VTSS_LB_H2	Host Loopback 2, 40-bit XAUI-PHY interface Mirror XAUI data
VTSS_LB_H3	Host Loopback 3, 64-bit PCS after the gearbox FF00 repeating IEEE PCS system loopback
VTSS_LB_H4	Host Loopback 4, 64-bit WIS FF00 repeating IEEE WIS system loopback
VTSS_LB_H5	Host Loopback 5, 1-bit SFP+ after SerDes Mirror XAUI data IEEE PMA system loopback
VTSS_LB_H6	Host Loopback 6, 32-bit XAUI-PHY interface Mirror XAUI data
VTSS_LB_L0	Line Loopback 0, 4-bit XAUI before SerDes Mirror SFP+ data
VTSS_LB_L1	Line Loopback 1, 4-bit XAUI after SerDes Mirror SFP+ data IEEE PHY-XS network loopback
VTSS_LB_L2	Line Loopback 2, 64-bit XGMII after FIFO Mirror SFP+ data
VTSS_LB_L3	Line Loopback 3, 64-bit PMA interface Mirror SFP+ data

Definition at line 1598 of file vtss\_phy\_10g\_api.h.

#### 6.30.4.29 vtss\_phy\_10g\_power\_t

enum `vtss_phy_10g_power_t`

10G Phy power setting

Enumerator

<code>VTSS_PHY_10G_POWER_ENABLE</code>	Enable Phy power for all sublayers
<code>VTSS_PHY_10G_POWER_DISABLE</code>	Disable Phy power for all sublayers

Definition at line 1699 of file vtss\_phy\_10g\_api.h.

#### 6.30.4.30 vtss\_phy\_10g\_failover\_mode\_t

enum `vtss_phy_10g_failover_mode_t`

10G Phy Failover Mode Setting

Enumerator

<code>VTSS_PHY_10G_PMA_TO_FROM_XAUI_NORMAL</code>	PMA_0/1 to XAUI_0/1. 8487: XAUI 0 to PMA 0
<code>VTSS_PHY_10G_PMA_TO_FROM_XAUI_CROSSED_SED</code>	PMA_0/1 to XAUI_1/0. 8487: XAUI 1 to PMA 0
<code>VTSS_PHY_10G_PMA_0_TO_FROM_XAUI_0_TO_XAUI_1</code>	PMA 0 to/from XAUI 0 and to XAUI 1
<code>VTSS_PHY_10G_PMA_0_TO_FROM_XAUI_1_TO_XAUI_0</code>	PMA 0 to/from XAUI 1 and to XAUI 0
<code>VTSS_PHY_10G_PMA_1_TO_FROM_XAUI_0_TO_XAUI_1</code>	PMA 1 to/from XAUI 0 and to XAUI 1. VSC8487:N/A
<code>VTSS_PHY_10G_PMA_1_TO_FROM_XAUI_1_TO_XAUI_0</code>	PMA 1 to/from XAUI 1 and to XAUI 0. VSC8487:N/A

Definition at line 1749 of file vtss\_phy\_10g\_api.h.

#### 6.30.4.31 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.

## 6.30.4.32 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.

## 6.30.4.33 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.

## 6.30.4.34 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.

## 6.30.4.35 vtss\_10g\_phy\_gpio\_t

enum [vtss\\_10g\\_phy\\_gpio\\_t](#)

GPIO configured mode.

GPIO configured mode

Enumerator

VTSS_10G_PHY_GPIO_NOT_INITIALIZED	This GPIO pin has been initialized by a call to API from application. registers contain power-up default value
VTSS_10G_PHY_GPIO_OUT	Output enabled
VTSS_10G_PHY_GPIO_IN	Input enabled
VTSS_10G_PHY_GPIO_WIS_INT	Output WIS interrupt channel 0 or 1 (depending on port_no) enabled
VTSS_10G_PHY_GPIO_1588_LOAD_SAVE	Input interrupt generated on falling edge Input interrupt generated on raising edge Input interrupt generated on raising and falling edge Input 1588 load/save function
VTSS_10G_PHY_GPIO_1588_1PPS_0	Output 1588 1PPS from channel 0 function
VTSS_10G_PHY_GPIO_1588_1PPS_1	Output 1588 1PPS from channel 1 function
VTSS_10G_PHY_GPIO_1588_1PPS_2	Output 1588 1PPS from channel 2 function
VTSS_10G_PHY_GPIO_1588_1PPS_3	Output 1588 1PPS from channel 3 function
VTSS_10G_PHY_GPIO_PCS_RX_FAULT	PCS_RX_FAULT (from channel 0 or 1) is transmitted on GPIO
VTSS_10G_PHY_GPIO_SET_I2C_MASTER	Used in communicating with I2C slave, like SPP+
VTSS_10G_PHY_GPIO_TX_ENABLE	Transmit enable , MALIBU
VTSS_10G_PHY_GPIO_LINE_PLL_STATUS	Line PLL Status , MALIBU
VTSS_10G_PHY_GPIO_HOST_PLL_STATUS	Host PLL Status , MALIBU
VTSS_10G_PHY_GPIO_RCOMP_BUSY	RCOMP busy Status , MALIBU
VTSS_10G_PHY_GPIO_CHAN_INT_0	Interrupt 0 from channel , MALIBU

## Enumerator

VTSS_10G_PHY_GPIO_CHAN_INT_1	Interrupt 1 from channel , MALIBU
VTSS_10G_PHY_GPIO_1588_INT	1588 Interrupt , MALIBU
VTSS_10G_PHY_GPIO_TS_FIFO_EMPTY	TS FIFO empty , MALIBU
VTSS_10G_PHY_GPIO_AGG_INT_0	Aggregated interrupt 0 , MALIBU
VTSS_10G_PHY_GPIO_AGG_INT_1	Aggregated interrupt 1 , MALIBU
VTSS_10G_PHY_GPIO_AGG_INT_2	Aggregated interrupt 2 , MALIBU
VTSS_10G_PHY_GPIO_AGG_INT_3	Aggregated interrupt 3 , MALIBU
VTSS_10G_PHY_GPIO_PLL_INT_0	Interrupt 0 from PLL , MALIBU
VTSS_10G_PHY_GPIO_PLL_INT_1	Interrupt 0 from PLL , MALIBU
VTSS_10G_PHY_GPIO_SET_I2C_SLAVE	I2C Slave set , MALIBU
VTSS_10G_PHY_GPIO_CRSS_INT	Cross Connect Interrupt , MALIBU
VTSS_10G_PHY_GPIO_LED	LED Setting , MALIBU
VTSS_10G_PHY_GPIO_DRIVE_LOW	GPIO output to LOW , MALIBU
VTSS_10G_PHY_GPIO_DRIVE_HIGH	GPIO output to HIGH , MALIBU

Definition at line 2306 of file vtss\_phy\_10g\_api.h.

6.30.4.36 `vtss_gpio_10g_gpio_intr_sgnl_t`

```
enum vtss_gpio_10g_gpio_intr_sgnl_t
```

GPIO internal signal types.

## Enumerator

VTSS_10G_GPIO_INTR_SGNL_I2C_MSTR_CLK↔_OUT	GPIO output I2C master data out
VTSS_10G_GPIO_INTR_SGNL_LED_TX	GPIO output I2C master clock out
VTSS_10G_GPIO_INTR_SGNL_LED_RX	LED transmit
VTSS_10G_GPIO_INTR_SGNL_RX_ALARM	LED receive
VTSS_10G_GPIO_INTR_SGNL_TX_ALARM	RX Alarm
VTSS_10G_GPIO_INTR_SGNL_HOST_LINK	TX Alarm
VTSS_10G_GPIO_INTR_SGNL_LINE_LINK	Host Link status
VTSS_10G_GPIO_INTR_SGNL_LINE_KR_8b10b↔_2GPIO	Line Link status
VTSS_10G_GPIO_INTR_SGNL_LINE_KR_10b_2↔_GPIO	KR 8b10b
VTSS_10G_GPIO_INTR_SGNL_ROSI_PULSE	KR 10b
VTSS_10G_GPIO_INTR_SGNL_ROSI_SDATA	ROSI Pulse
VTSS_10G_GPIO_INTR_SGNL_ROSI_SCLK	ROSI sdata
VTSS_10G_GPIO_INTR_SGNL_TOSI_PULSE	ROSI sclock
VTSS_10G_GPIO_INTR_SGNL_TOSI_SCLK	TOSI Pulse
VTSS_10G_GPIO_INTR_SGNL_LINE_PCS1G_LINK	TOSI sclock
VTSS_10G_GPIO_INTR_SGNL_LINE_PCS_RX_↔_STAT	Line PCS1G link status

## Enumerator

VTSS_10G_GPIO_INTR_SGNL_CLIENT_PCS1G_LINK	Line PCS RX link status
VTSS_10G_GPIO_INTR_SGNL_HOST_PCS_RX_STAT	Client PCS1G link status
VTSS_10G_GPIO_INTR_SGNL_HOST_SD10G_IB_SIG	Host PCS RX status
VTSS_10G_GPIO_INTR_SGNL_LINE_SD10G_IB_SIG	Host SERDES 10G 1B signal
VTSS_10G_GPIO_INTR_SGNL_HPCS_INTR	Line SERDES 10G 1B signal
VTSS_10G_GPIO_INTR_SGNL_LPCS_INTR	HPCS interrupt
VTSS_10G_GPIO_INTR_SGNL_CLIENT_PCS1G_INTR	LPCS interrupt
VTSS_10G_GPIO_INTR_SGNL_LINE_PCS1G_INTR	Client PCS1G interrupt
VTSS_10G_GPIO_INTR_SGNL_WIS_INT0	Line PCS1G interrupt
VTSS_10G_GPIO_INTR_SGNL_HOST_PMA_INT	WIS interrupt 0
VTSS_10G_GPIO_INTR_SGNL_LINE_PMA_INT	Host PMA interrupt
VTSS_10G_GPIO_INTR_SGNL_DATA_ACT_TX	Line PMA interrupt
VTSS_10G_GPIO_INTR_SGNL_DATA_ACT_RX	TX data activity
VTSS_10G_GPIO_INTR_SGNL_HDATA_ACT_TX	RX data activity
VTSS_10G_GPIO_INTR_SGNL_HDATA_ACT_RX	Host TX data activity
	Host RX data activity
VTSS_10G_GPIO_INTR_SGNL_XGMII_PAUS_EGR	
VTSS_10G_GPIO_INTR_SGNL_XGMII_PAUS_ING	XGMI pause egress
VTSS_10G_GPIO_INTR_SGNL_RX_PCS_PAUS	XGMI pause ingress
VTSS_10G_GPIO_INTR_SGNL_TX_PCS_PAUS	PCS RX Pause
VTSS_10G_GPIO_INTR_SGNL_RX_WIS_PAUS	PCS TX Pause
VTSS_10G_GPIO_INTR_SGNL_TX_WIS_PAUS	WIS RX Pause
VTSS_10G_GPIO_INTR_SGNL_ETH_CHAN_DIS	WIS TX Pause
VTSS_10G_GPIO_INTR_SGNL_MACSEC_1588_SFD_LANE	Ethernet channel disable
	MACSEC,1588 SFD lane
VTSS_10G_GPIO_INTR_SGNL_LINE_S_TX_FAULT	
VTSS_10G_GPIO_INTR_SGNL_LPCS1G_LATEN_CY0_OR_EWIS_BIT0	TX fault
VTSS_10G_GPIO_INTR_SGNL_LPCS1G_LATEN_CY1_OR_EWIS_BIT1	LPCS1G latency 0 in case of 1G mode or EWIS BIT 0
VTSS_10G_GPIO_INTR_SGNL_LPCS1G_CHAR_POS0_OR_EWIS_BIT2	LPCS1G latency 0 in case of 1G mode or EWIS BIT 1
VTSS_10G_GPIO_INTR_SGNL_LPCS1G_CHAR_POS1_OR_EWIS_WORD0	LPCS1G Char pos 0 in case of 1G mode or EWIS BIT 2
VTSS_10G_GPIO_INTR_SGNL_LPCS1G_CHAR_POS2_OR_EWIS_WORD1	LPCS1G Char pos 1 in case of 1G mode or EWIS word 0
VTSS_10G_GPIO_INTR_SGNL_LPCS1G_CHAR_POS3_OR_EWIS_WORD2	LPCS1G Char pos 2 in case of 1G mode or EWIS word 1
VTSS_10G_GPIO_INTR_SGNL_MACSEC_IGR_PRED_VAR0	LPCS1G Char pos 3 in case of 1G mode or EWIS word 2
VTSS_10G_GPIO_INTR_SGNL_MACSEC_IGR_PRED_VAR1	Macsec ingress predictor var 0
VTSS_10G_GPIO_INTR_SGNL_KR_ACTV_2GPIO	Macsec ingress predictor var 1
VTSS_10G_GPIO_INTR_SGNL_DFT_TX_2GPIO	KR activity

## Enumerator

VTSS_10G_GPIO_INTR_SGNL_RESERVED	DFT transmit
VTSS_10G_GPIO_INTR_SGNL_EXE_LST_2GPIO_0	Reserved for future use
VTSS_10G_GPIO_INTR_SGNL_EXE_LST_2GPIO_1	EXE LST to GPIO 0
VTSS_10G_GPIO_INTR_SGNL_EXE_LST_2GPIO_2	EXE LST to GPIO 1
VTSS_10G_GPIO_INTR_SGNL_EXE_LST_2GPIO_3	EXE LST to GPIO 2
VTSS_10G_GPIO_INTR_SGNL_EXE_LST_2GPIO_4	EXE LST to GPIO 3
VTSS_10G_GPIO_INTR_SGNL_LINK_HCD_2GPIO_0	EXE LST to GPIO 4
VTSS_10G_GPIO_INTR_SGNL_LINK_HCD_2GPIO_1	Link HCD to GPIO 0
VTSS_10G_GPIO_INTR_SGNL_LINK_HCD_2GPIO_2	Link HCD to GPIO 1
VTSS_10G_GPIO_INTR_SGNL_ETH_1G_ENA	Link HCD to GPIO 2
VTSS_10G_GPIO_INTR_SGNL_H_KR_8b10b_2GPIO	Ethernet 1G enable
VTSS_10G_GPIO_INTR_SGNL_H_KR_10Gb_2GPIO	KR 8b10b to GPIO
VTSS_10G_GPIO_INTR_SGNL_H_KR_ACTV_2GPIO	KR10Gb to GPIO
VTSS_10G_GPIO_INTR_SGNL_NONE	KR activity to GPIO

Definition at line 2347 of file vtss\_phy\_10g\_api.h.

## 6.30.4.37 vtss\_gpio\_10g\_chan\_intrpt\_t

```
enum vtss_gpio_10g_chan_intrpt_t
```

GPIO Channel level interrupts.

Internal signals supported on Malibu

## Enumerator

VTSS_10G_GPIO_INTRPT_WIS1	WIS interrupt 0
VTSS_10G_GPIO_INTRPT_LPCS10G	WIS interrupt 1
VTSS_10G_GPIO_INTRPT_HPCS10G	LPCS 10G interrupt
VTSS_10G_GPIO_INTRPT_LPCS1G	HPCS 10G interrupt
VTSS_10G_GPIO_INTRPT_HPCS1G	LPCS 1G interrupt
VTSS_10G_GPIO_INTRPT_MSEC_EGR	HPCS 1G interrupt
VTSS_10G_GPIO_INTRPT_MSEC_IGR	Macsec Egress interrupt
VTSS_10G_GPIO_INTRPT_LMAC	Macsec Ingress interrupt
VTSS_10G_GPIO_INTRPT_HMAC	Line MAC interrupt
VTSS_10G_GPIO_INTRPT_FCBUF	Host MAC interrupt

## Enumerator

VTSS_10G_GPIO_INTRPT_LIGR_FIFO	FC Buffer interrupt
VTSS_10G_GPIO_INTRPT_LEGR_FIFO	Line ingress FIFO interrupt
VTSS_10G_GPIO_INTRPT_HEGR_FIFO	Line egress FIFO interrupt
VTSS_10G_GPIO_INTRPT_LPMA	Host egress FIFO interrupt
VTSS_10G_GPIO_INTRPT_HPMA	Line PMA interrupt

Definition at line 2419 of file vtss\_phy\_10g\_api.h.

## 6.30.4.38 vtss\_gpio\_10g\_aggr\_intrpt\_t

```
enum vtss_gpio_10g_aggr_intrpt_t
```

GPIO Channel level interrupts.

## Channel Level Interrupts

## Enumerator

VTSS_10G_GPIO_AGGR_INTRPT_CH0_INTR1_EN	CH0_INTR0_EN
VTSS_10G_GPIO_AGGR_INTRPT_CH1_INTR0_EN	CH0_INTR1_EN
VTSS_10G_GPIO_AGGR_INTRPT_CH1_INTR1_EN	CH1_INTR0_EN
VTSS_10G_GPIO_AGGR_INTRPT_CH2_INTR0_EN	CH1_INTR1_EN
VTSS_10G_GPIO_AGGR_INTRPT_CH2_INTR1_EN	CH2_INTR0_EN
VTSS_10G_GPIO_AGGR_INTRPT_CH3_INTR0_EN	CH2_INTR1_EN
VTSS_10G_GPIO_AGGR_INTRPT_CH3_INTR1_EN	CH3_INTR0_EN
VTSS_10G_GPIO_AGGR_INTRPT_IP1588_0_INTR0_EN	CH3_INTR1_EN
VTSS_10G_GPIO_AGGR_INTRPT_IP1588_0_INTR1_EN	IP1588_0_INTR0_EN
VTSS_10G_GPIO_AGGR_INTRPT_IP1588_0_INTR2_EN	IP1588_0_INTR1_EN
VTSS_10G_GPIO_AGGR_INTRPT_IP1588_0_INTR3_EN	IP1588_0_INTR2_EN
VTSS_10G_GPIO_AGGR_INTRPT_IP1588_1_INTR0_EN	IP1588_0_INTR3_EN
VTSS_10G_GPIO_AGGR_INTRPT_IP1588_1_INTR1_EN	TS_FIFO empty channel 0
VTSS_10G_GPIO_AGGR_INTRPT_IP1588_1_INTR2_EN	TS_FIFO empty channel 1
VTSS_10G_GPIO_AGGR_INTRPT_IP1588_1_INTR3_EN	TS_FIFO empty channel 2
VTSS_10G_GPIO_AGGR_INTRPT_LCPLL_0_INTR_EN	TS_FIFO empty channel 3
VTSS_10G_GPIO_AGGR_INTRPT_LCPLL_1_INTR_EN	LCPLL_0_INTR_EN
VTSS_10G_GPIO_AGGR_INTRPT_EXP4_INTR_EN	LCPLL_1_INTR_EN
VTSS_10G_GPIO_AGGR_INTRPT_CLK_MUX_INTR_EN	EXP4_INTR_EN
VTSS_10G_GPIO_AGGR_INTRPT_GPIO_INTR_EN	CLK_MUX_INTR_EN

Definition at line 2442 of file vtss\_phy\_10g\_api.h.

## 6.30.4.39 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.

### 6.30.5 Function Documentation

#### 6.30.5.1 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.

#### 6.30.5.2 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.

**6.30.5.3 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.

**6.30.5.4 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.

**6.30.5.5 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.

**6.30.5.6 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.

**6.30.5.7 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.

**6.30.5.8 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.

**6.30.5.9 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.

**6.30.5.10 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.

**6.30.5.11 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.

**6.30.5.12 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.

**6.30.5.13 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.

**6.30.5.14 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.

**6.30.5.15 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.

**6.30.5.16 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.

**6.30.5.17 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.

**6.30.5.18 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.

### 6.30.5.19 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.

### 6.30.5.20 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.

### 6.30.5.21 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.

**6.30.5.22 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.

**6.30.5.23 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.

**6.30.5.24 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.

**6.30.5.25 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.

**6.30.5.26 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.

**6.30.5.27 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.

### 6.30.5.28 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.

### 6.30.5.29 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.

### 6.30.5.30 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.

### 6.30.5.31 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.

### 6.30.5.32 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.

### 6.30.5.33 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.

### 6.30.5.34 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.

**6.30.5.35 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.

**6.30.5.36 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.

**6.30.5.37 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.

**6.30.5.38 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.

**6.30.5.39 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.

**6.30.5.40 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.

**6.30.5.41 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.

**6.30.5.42 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.

### 6.30.5.43 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.

### 6.30.5.44 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.

### 6.30.5.45 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.

**6.30.5.46 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.

**6.30.5.47 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

**6.30.5.48 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.

**6.30.5.49 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.

**6.30.5.50 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.

**6.30.5.51 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.

### 6.30.5.52 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.

### 6.30.5.53 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.

### 6.30.5.54 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.

**6.30.5.55 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.

**6.30.5.56 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.

**6.30.5.57 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.

**6.30.5.58 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.

### 6.30.5.59 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.

### 6.30.5.60 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.

### 6.30.5.61 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.

### 6.30.5.62 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.

### 6.30.5.63 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.

### 6.30.5.64 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.

### 6.30.5.65 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.

### 6.30.5.66 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.

### 6.30.5.67 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.

### 6.30.5.68 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.

### 6.30.5.69 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.

**6.30.5.70 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.

**6.30.5.71 vtss\_phy\_10g\_pkt\_mon\_conf()**

```
vtss_rc vtss_phy_10g_pkt_mon_conf (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    BOOL ts_rd,
    vtss_phy_10g_pkt_mon_conf_t *const conf,
    vtss_phy_10g_timestamp_val_t *const conf_ts )
```

Set Packet Monitor Configuration.

**Parameters**

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>ts_rd</i>	[IN] Flag to indicate that timestamp fifo is also to be read.
<i>conf</i>	Packet monitor configuration
<i>conf</i> → <i>ts</i>	[OUT] Timestamp value array.

**Returns**

VTSS\_RC\_OK on success.

VTSS\_RC\_ERROR on error.

**6.30.5.72 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.

**6.30.5.73 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.

**6.30.5.74 vtss\_phy\_10g\_gpio\_mode\_set()**

```
vtss_rc vtss_phy_10g_gpio_mode_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_gpio_10g_no_t gpio_no,
    const vtss_gpio_10g_gpio_mode_t *const mode )
```

Set GPIO mode. There is only one set of GPIO per PHY chip - not per port.

**Parameters**

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number that identify the PHY chip.
<i>gpio_no</i>	[IN] GPIO pin number < VTSS_10G_PHY_GPIO_MAX.
<i>mode</i>	[IN] GPIO mode.

**Returns**

Return code.

**6.30.5.75 vtss\_phy\_10g\_gpio\_mode\_get()**

```
vtss_rc vtss_phy_10g_gpio_mode_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_gpio_10g_no_t gpio_no,
    vtss_gpio_10g_gpio_mode_t *const mode )
```

Get GPIO mode.

**Parameters**

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number that identify the PHY chip.
<i>gpio_no</i>	[IN] GPIO pin number.
<i>mode</i>	[OUT] GPIO mode.

Generated by Doxygen

**Returns**

Return code.

**6.30.5.76 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.

**6.30.5.77 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.

### 6.30.5.78 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.

### 6.30.5.79 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.

### 6.30.5.80 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.

**6.30.5.81 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.

**6.30.5.82 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.

**6.30.5.83 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.

**6.30.5.84 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.

### 6.30.5.85 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.

### 6.30.5.86 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.

### 6.30.5.87 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

**6.30.5.88 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.

**6.30.5.89 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.

**6.30.5.90 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.

**6.30.5.91 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.

### 6.30.5.92 vtss\_phy\_10g\_i2c\_read()

```
vtss_rc vtss_phy_10g_i2c_read (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const u16 addr,
    u16 * value )
```

read from i2c device

#### Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>addr</i>	[IN] Addr of the SFP ROM register
<i>value</i>	[OUT] Return Value of the register

#### Returns

Return code.

### 6.30.5.93 vtss\_phy\_10g\_i2c\_write()

```
vtss_rc vtss_phy_10g_i2c_write (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const u16 addr,
    const u16 * value )
```

Write to i2c device.

#### Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>addr</i>	[IN] Addr of the SFP ROM register
<i>value</i>	[IN] value to be written to register

#### Returns

Return code.

### 6.30.5.94 vtss\_phy\_10g\_get\_user\_data()

```
vtss_rc vtss_phy_10g_get_user_data (
    const vtss_inst_t inst,
```

```
const vtss_port_no_t port_no,
void ** user_data )
```

Gets generic pointer in vtss\_state structure.

#### Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>user_data</i>	[OUT] Gets value in generic pointer

#### Returns

Return code.

## 6.31 vtss\_api/include/vtss\_phy\_api.h File Reference

PHY API.

```
#include <vtss/api/types.h>
#include <vtss/api/port.h>
#include <vtss/api/phy.h>
#include <vtss_misc_api.h>
```

## Data Structures

- struct [vtss\\_phy\\_led\\_mode\\_select\\_t](#)  
*LED model selection.*
- struct [vtss\\_phy\\_type\\_t](#)  
*Phy type information.*
- struct [vtss\\_phy\\_rgmii\\_conf\\_t](#)  
*PHY RGMII configuration.*
- struct [vtss\\_phy\\_tbi\\_conf\\_t](#)  
*PHY TBI configuration.*
- struct [vtss\\_phy\\_reset\\_conf\\_t](#)  
*PHY reset structure.*
- struct [vtss\\_phy\\_forced\\_t](#)  
*PHY forced mode configuration.*
- struct [vtss\\_phy\\_aneg\\_t](#)  
*PHY auto negotiation advertisement.*
- struct [vtss\\_phy\\_mac\\_serdes\\_pcs\\_ctrl\\_t](#)  
*PHY MAC SerDes PCS Control, Reg16E3.*
- struct [vtss\\_phy\\_media\\_serdes\\_pcs\\_ctrl\\_t](#)  
*PHY MEDIA SerDes PCS Control, Reg23E3.*
- struct [vtss\\_phy\\_conf\\_t](#)  
*PHY configuration.*
- struct [vtss\\_phy\\_conf\\_1g\\_t](#)  
*PHY 1G configuration.*

- struct `vtss_phy_status_1g_t`  
*PHY 1G status.*
- struct `vtss_phy_power_conf_t`  
*PHY power configuration.*
- struct `vtss_phy_power_status_t`  
*PHY power status.*
- struct `vtss_phy_clock_conf_t`  
*PHY clock configuration.*
- struct `vtss_phy_veriphy_result_t`  
*VeriPHY result.*
- struct `vtss_phy_eee_conf_t`  
*EEE configuration.*
- struct `vtss_phy_enhanced_led_control_t`  
*enhanced LED control*
- struct `vtss_phy_statistic_t`  
*Phy statistic information.*
- struct `vtss_phy_loopback_t`  
*1G Phy loopbacks*
- struct `vtss_wol_mac_addr_t`  
*Structure for Wake-On-LAN MAC Address.*
- struct `vtss_secure_on_passwd_t`  
*Structure for Wake-On-LAN Secure-On Password.*
- struct `vtss_phy_wol_conf_t`  
*Structure for Get/Set Wake-On-LAN configuration.*
- struct `vtss_rcpll_status_t`  
*Structure for Get PHY RC-PLL status.*
- struct `vtss_lcpll_status_t`  
*Structure for Get PHY LC-PLL status.*

## Macros

- #define MAX\_CFG\_BUF\_SIZE 38
- #define MAX\_STAT\_BUF\_SIZE 8
- #define VTSS\_PHY\_POWER\_ACTIPHYS\_BIT 0
- #define VTSS\_PHY\_POWER\_DYNAMIC\_BIT 1
- #define VTSS\_PHY\_ACTIPHYS\_PWR 100
- #define VTSS\_PHY\_LINK\_DOWN\_PWR 200
- #define VTSS\_PHY\_LINK\_UP\_FULL\_PWR 400
- #define VTSS\_PHY\_RECov\_CLK1 0  
*PHY active clock out.*
- #define VTSS\_PHY\_RECov\_CLK2 1
- #define VTSS\_PHY\_RECov\_CLK\_NUM 2
- #define VTSS\_PHY\_PAGE\_STANDARD 0x0000
- #define VTSS\_PHY\_PAGE\_EXTENDED 0x0001
- #define VTSS\_PHY\_PAGE\_EXTENDED\_2 0x0002
- #define VTSS\_PHY\_PAGE\_EXTENDED\_3 0x0003
- #define VTSS\_PHY\_PAGE\_EXTENDED\_4 0x0004
- #define VTSS\_PHY\_PAGE\_GPIO 0x0010
- #define VTSS\_PHY\_PAGE\_1588 0x1588
- #define VTSS\_PHY\_PAGE\_MACSEC 0x0004
- #define VTSS\_PHY\_PAGE\_TEST 0x2A30

- #define VTSS\_PHY\_PAGE\_TR 0x52B5
- #define VTSS\_PHY\_PAGE\_0x2DAF 0x2DAF
- #define VTSS\_PHY\_REG\_STANDARD (VTSS\_PHY\_PAGE\_STANDARD<<5)
- #define VTSS\_PHY\_REG\_EXTENDED (VTSS\_PHY\_PAGE\_EXTENDED<<5)
- #define VTSS\_PHY\_REG\_GPIO (VTSS\_PHY\_PAGE\_GPIO<<5)
- #define VTSS\_PHY\_REG\_TEST (VTSS\_PHY\_PAGE\_TEST<<5)
- #define VTSS\_PHY\_REG\_TR (VTSS\_PHY\_PAGE\_TR<<5)
- #define VTSS\_PHY\_LINK\_LOS\_EV (1 << 0)
- #define VTSS\_PHY\_LINK\_FFAIL\_EV (1 << 1)
- #define VTSS\_PHY\_LINK\_AMS\_EV (1 << 2)
- #define VTSS\_PHY\_LINK\_SPEED\_STATE\_CHANGE\_EV (1 << 3)
- #define VTSS\_PHY\_LINK\_FDX\_STATE\_CHANGE\_EV (1 << 4)
- #define VTSS\_PHY\_LINK\_AUTO\_NEG\_ERROR\_EV (1 << 5)
- #define VTSS\_PHY\_LINK\_AUTO\_NEG\_COMPLETE\_EV (1 << 6)
- #define VTSS\_PHY\_LINK\_INLINE\_POW\_DEV\_DETECT\_EV (1 << 7)
- #define VTSS\_PHY\_LINK\_SYMBOL\_ERR\_INT\_EV (1 << 8)
- #define VTSS\_PHY\_LINK\_TX\_FIFO\_OVERFLOW\_INT\_EV (1 << 9)
- #define VTSS\_PHY\_LINK\_RX\_FIFO\_OVERFLOW\_INT\_EV (1 << 10)
- #define VTSS\_PHY\_LINK\_FALSE\_CARRIER\_INT\_EV (1 << 11)
- #define VTSS\_PHY\_LINK\_LINK\_SPEED\_DS\_DETECT\_EV (1 << 12)
- #define VTSS\_PHY\_LINK\_MASTER\_SLAVE\_RES\_ERR\_EV (1 << 13)
- #define VTSS\_PHY\_LINK\_RX\_ER\_INT\_EV (1 << 14)
- #define VTSS\_PHY\_LINK\_EXTENDED\_REG\_INT\_EV (1 << 15)
- #define VTSS\_PHY\_LINK\_WAKE\_ON\_LAN\_INT\_EV (1 << 16)
- #define VTSS\_PHY\_LINK\_EXT\_EEE\_WAKE\_ERR\_EV (1 << 17)
- #define VTSS\_PHY\_LINK\_EXT\_EEE\_WAIT\_TS\_EV (1 << 18)
- #define VTSS\_PHY\_LINK\_EXT\_EEE\_WAIT\_RX\_TQ\_EV (1 << 19)
- #define VTSS\_PHY\_LINK\_EXT\_EEE\_LINKFAIL\_EV (1 << 20)
- #define VTSS\_PHY\_LINK\_EXT\_RR\_SW\_COMPL\_EV (1 << 21)
- #define VTSS\_PHY\_LINK\_EXT\_MACSEC\_HOST\_MAC\_EV (1 << 22)
- #define VTSS\_PHY\_LINK\_EXT\_MACSEC\_LINE\_MAC\_EV (1 << 23)
- #define VTSS\_PHY\_LINK\_EXT\_MACSEC\_FC\_BUFF\_EV (1 << 24)
- #define VTSS\_PHY\_LINK\_EXT\_MACSEC\_INGRESS\_EV (1 << 25)
- #define VTSS\_PHY\_LINK\_EXT\_MACSEC\_EGRESS\_EV (1 << 26)
- #define VTSS\_PHY\_LINK\_EXT\_MEM\_INT\_RING\_EV (1 << 27)
- #define MAX\_WOL\_MAC\_ADDR\_SIZE 6
- #define MAX\_WOL\_PASSWD\_SIZE 6
- #define MIN\_WOL\_PASSWD\_SIZE 4

## Typedefs

- typedef u16 vtss\_phy\_recov\_clk\_t
- typedef u8 vtss\_phy\_led\_intensity
 

*PHY led intensity.*
- typedef u32 vtss\_phy\_event\_t
 

*PHY interrupt event type.*

## Enumerations

- enum `vtss_phy_part_number_t` {
   
`VTSS_PHY_TYPE_NONE` = 0, `VTSS_PHY_TYPE_8201` = 8201, `VTSS_PHY_TYPE_8204` = 8204, `VTSS_PHY_TYPE_8211` = 8211,
 `VTSS_PHY_TYPE_8221` = 8221, `VTSS_PHY_TYPE_8224` = 8224, `VTSS_PHY_TYPE_8234` = 8234, `VTSS_PHY_TYPE_8244` = 8244,
 `VTSS_PHY_TYPE_8538` = 8538, `VTSS_PHY_TYPE_8558` = 8558, `VTSS_PHY_TYPE_8574` = 8574, `VTSS_PHY_TYPE_8504` = 8504,
 `VTSS_PHY_TYPE_8572` = 8572, `VTSS_PHY_TYPE_8552` = 8552, `VTSS_PHY_TYPE_8501` = 8501, `VTSS_PHY_TYPE_8502` = 8502,
 `VTSS_PHY_TYPE_7435` = 7435, `VTSS_PHY_TYPE_8658` = 8658, `VTSS_PHY_TYPE_8601` = 8601, `VTSS_PHY_TYPE_8641` = 8641,
 `VTSS_PHY_TYPE_7385` = 7385, `VTSS_PHY_TYPE_7388` = 7388, `VTSS_PHY_TYPE_7389` = 7389, `VTSS_PHY_TYPE_7390` = 7390,
 `VTSS_PHY_TYPE_7395` = 7395, `VTSS_PHY_TYPE_7398` = 7398, `VTSS_PHY_TYPE_7500` = 7500, `VTSS_PHY_TYPE_7501` = 7501,
 `VTSS_PHY_TYPE_7502` = 7502, `VTSS_PHY_TYPE_7503` = 7503, `VTSS_PHY_TYPE_7504` = 7504, `VTSS_PHY_TYPE_7505` = 7505,
 `VTSS_PHY_TYPE_7506` = 7506, `VTSS_PHY_TYPE_7507` = 7507, `VTSS_PHY_TYPE_8634` = 8634, `VTSS_PHY_TYPE_8664` = 8664,
 `VTSS_PHY_TYPE_8512` = 8512, `VTSS_PHY_TYPE_8522` = 8522, `VTSS_PHY_TYPE_7420` = 7420, `VTSS_PHY_TYPE_8582` = 8582,
 `VTSS_PHY_TYPE_8584` = 8584, `VTSS_PHY_TYPE_8575` = 8575, `VTSS_PHY_TYPE_8564` = 8564, `VTSS_PHY_TYPE_8562` = 8562,
 `VTSS_PHY_TYPE_8586` = 8586, `VTSS_PHY_TYPE_8514` = 8514 }
   
*PHY part ids supported.*
- enum `vtss_phy_led_mode_t` {
   
`LINK_ACTIVITY`, `LINK100_ACTIVITY`, `LINK100_ACTIVITY`, `LINK10_ACTIVITY`,
 `LINK100_1000_ACTIVITY`, `LINK10_1000_ACTIVITY`, `LINK10_100_ACTIVITY`, `LINK100BASE_FX_1000BASE_X_ACTIVITY`,
 `DUPLEX_COLLISION`, `COLLISION_ACTIVITY`, `BASE100_FX_1000BASE_X_FIBER_ACTIVITY`,
 `AUTONEGOTIATION_FAULT`, `LINK100BASE_X_ACTIVITY`, `LINK100BASE_FX_ACTIVITY`, `BASE100_ACTIVITY`,
 `BASE100_FX_ACTIVITY`, `FORCE_LED_OFF`, `FORCE_LED_ON`, `FAST_LINK_FAIL` }
   
*PHY LED modes.*
- enum `vtss_phy_led_number_t` { `LED0`, `LED1`, `LED2`, `LED3` }
   
*List of LED pins per port.*
- enum `vtss_phy_media_interface_t` {
   
`VTSS_PHY_MEDIA_IF_CU`, `VTSS_PHY_MEDIA_IF_SFP_PASSTHRU`, `VTSS_PHY_MEDIA_IF_FL_1000BX`,
 `VTSS_PHY_MEDIA_IF_FL_100FX`,
 `VTSS_PHY_MEDIA_IF_AMS CU_PASSTHRU`, `VTSS_PHY_MEDIA_IF_AMS FI_PASSTHRU`, `VTSS_PHY_MEDIA_IF_AMS`
`VTSS_PHY_MEDIA_IF_AMS FI_1000BX`,
 `VTSS_PHY_MEDIA_IF_AMS CU_100FX`, `VTSS_PHY_MEDIA_IF_AMS FI_100FX` }
   
*PHY media interface type.*
- enum `vtss_phy_mdi_t` { `VTSS_PHY_MDIX_AUTO`, `VTSS_PHY_MDI`, `VTSS_PHY_MDIX` }
   
*PHY media interface type.*
- enum `rgmii_skew_delay_psec_t` {
   
`rgmii_skew_delay_200_psec` = 200, `rgmii_skew_delay_800_psec` = 800, `rgmii_skew_delay_1100_psec` =
 1100, `rgmii_skew_delay_1700_psec` = 1700,
 `rgmii_skew_delay_2000_psec` = 2000, `rgmii_skew_delay_2300_psec` = 2300, `rgmii_skew_delay_2600_psec` =
 2600, `rgmii_skew_delay_3400_psec` = 3400 }
   
*RGMII skew values.*
- enum `vtss_phy_forced_reset_t` { `VTSS_PHY_FORCE_RESET` = 0, `VTSS_PHY_NOFORCE_RESET` = 1 }
   
*PHY forced reset interface type.*
- enum `vtss_phy_pkt_mode_t` { `VTSS_PHY_PKT_MODE_IEEE_1_5_KB`, `VTSS_PHY_PKT_MODE_JUMBO_9_KB`,
 `VTSS_PHY_PKT_MODE_JUMBO_12_KB` }
   
*PHY packet mode configuration.*

- enum `vtss_phy_mode_t`{ `VTSS_PHY_MODE_ANEG`, `VTSS_PHY_MODE_FORCED`, `VTSS_PHY_MODE_POWER_DOWN` }

*PHY mode.*

- enum `vtss_phy_fast_link_fail_t`{ `VTSS_PHY_FAST_LINK_FAIL_ENABLE` = 1, `VTSS_PHY_FAST_LINK_FAIL_DISABLE` }

*PHY fast link failure pin enable/disable.*

- enum `vtss_phy_sigdet_polarity_t`{ `VTSS_PHY_SIGDET_POLARITY_ACT_LOW` = 1, `VTSS_PHY_SIGDET_POLARITY_ACT_HIGH` }

*PHY Sigdet pin polarity configuration.*

- enum `vtss_phy_unidirectional_t`{ `VTSS_PHY_UNIDIRECTIONAL_DISABLE` = 0, `VTSS_PHY_UNIDIRECTIONAL_ENABLE` }

*PHY Unidirectional enable/disable.*

- enum `vtss_phy_mac_serdes_pcs_sgmii_pre` { `VTSS_PHY_MAC_SERD_PCS_SGMII_IN_PRE_NONE` = 0, `VTSS_PHY_MAC_SERD_PCS_SGMII_IN_PRE_ONE` = 1, `VTSS_PHY_MAC_SERD_PCS_SGMII_IN_PRE_TWO` = 2, `VTSS_PHY_MAC_SERD_PCS_SGMII_IN_PRE_RSVD` = 3 }

*PHY MEDIA SerDes PCS Remote Fault Indication, See Clause 37, Table 37-3.*

- enum `vtss_phy_media_rem_fault_t` { `VTSS_PHY_MEDIA_SERD_PCS_Rem_Fault_No_Error` = 0, `VTSS_PHY_MEDIA_SERD_PCS_Rem_Fault_Offline` = 1, `VTSS_PHY_MEDIA_SERD_PCS_Rem_Fault_Link_Fail` = 2, `VTSS_PHY_MEDIA_SERD_PCS_Rem_Fault_ANEG_Error` = 3 }

*PHY MEDIA SerDes PCS Remote Fault Indication, See Clause 37, Table 37-3.*

- enum `vtss_phy_media_force_ams_sel_t` { `VTSS_PHY_MEDIA_FORCE_AMS_SELECTION_NORMAL` = 0, `VTSS_PHY_MEDIA_FORCE_AMS_SELECTION_SERDES` = 1, `VTSS_PHY_MEDIA_FORCE_AMS_SELECTION_COPPER` = 2 }

*PHY AMS Force configuration.*

- enum `vtss_phy_clk_source_t` { `VTSS_PHY_CLK_DISABLED`, `VTSS_PHY_SERDES_MEDIA`, `VTSS_PHY_COPPER_MEDIA`, `VTSS_PHY_TCLK_OUT`, `VTSS_PHY_LOCAL_XTAL` }

*PHY clock sources.*

- enum `vtss_phy_freq_t` { `VTSS_PHY_FREQ_25M`, `VTSS_PHY_FREQ_125M`, `VTSS_PHY_FREQ_3125M` }

*PHY clock frequencies.*

- enum `vtss_phy_clk_squelch` { `VTSS_PHY_CLK_SQUELCH_MAX` = 0, `VTSS_PHY_CLK_SQUELCH_MED` = 1, `VTSS_PHY_CLK_SQUELCH_MIN` = 2, `VTSS_PHY_CLK_SQUELCH_NONE` = 3 }

*PHY clock squelch levels.*

- enum `vtss_phy_gpio_mode_t` { `VTSS_PHY_GPIO_ALT_0` = 0, `VTSS_PHY_GPIO_ALT_1` = 1, `VTSS_PHY_GPIO_ALT_2` = 2, `VTSS_PHY_GPIO_OUT` = 3, `VTSS_PHY_GPIO_IN` = 4 }

*GPIO pin operating mode.*

- enum `vtss_eee_mode_t` { `EEE_DISABLE`, `EEE_ENABLE`, `EEE_REG_UPDATE` }

*EEE mode.*

- enum `lb_type` { `VTSS_LB_1G_NONE`, `VTSS_LB_FAR_END`, `VTSS_LB_NEAR_END` }

*Internal loop-back type.*

- enum `vtss_wol_passwd_len_type_t` { `VTSS_WOL_PASSWD_LEN_4` = 4, `VTSS_WOL_PASSWD_LEN_6` = 6 }

*Structure for Wake-On-LAN Password Length configuration.*

- enum `vtss_fefi_mode_t` { `VTSS_100FX_FEFI_NORMAL` = 0, `VTSS_100FX_FEFI_FORCE_SUPPRESS` = 2, `VTSS_100FX_FEFI_FORCE_ENABLE` = 3 }

*Far-End Failure Indication modes These settings map to PHY Reg23E3, bits 0 and 1.*

## Functions

- `vtss_rc vtss_phy_pre_reset (const vtss_inst_t inst, const vtss_port_no_t port_no)`  
*Must be call previous to port PHY Reset (vtss\_phy\_reset).*
- `vtss_rc vtss_phy_post_reset (const vtss_inst_t inst, const vtss_port_no_t port_no)`  
*Must be call after port PHY Reset (vtss\_phy\_reset).*
- `vtss_rc vtss_phy_pre_system_reset (const vtss_inst_t inst, const vtss_port_no_t port_no)`  
*Must be call before a system reset.*
- `vtss_rc vtss_phy_reset (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_reset_conf_t *const conf)`  
*Reset PHY.*
- `vtss_rc vtss_phy_reset_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_reset_conf_t *conf)`  
*Get reset configuration.*
- `vtss_rc vtss_phy_chip_temp_get (const vtss_inst_t inst, const vtss_port_no_t port_no, i16 *const temp)`  
*Get chip temperature.*
- `vtss_rc vtss_phy_chip_temp_init (const vtss_inst_t inst, const vtss_port_no_t port_no)`  
*Init. chip temperature.*
- `vtss_rc vtss_phy_conf_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_conf_t *const conf)`  
*Get PHY configuration.*
- `vtss_rc vtss_phy_conf_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_conf_t *const conf)`  
*Set PHY configuration.*
- `vtss_rc vtss_phy_status_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_port_status_t *const status)`  
*Get PHY status.*
- `vtss_rc vtss_phy_cl37_lp_abil_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_port_status_t *const status)`  
*Get Clause37 Link pArtnr's ability.*
- `vtss_rc vtss_phy_id_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_type_t *phy_id)`  
*Get the PHY type/id. The the Phy ID - can only be used after setting the operating mode.*
- `vtss_rc vtss_phy_conf_1g_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_conf_1g_t *const conf)`  
*Get PHY 1G configuration.*
- `vtss_rc vtss_phy_conf_1g_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_conf_1g_t *const conf)`  
*Set PHY 1G configuration.*
- `vtss_rc vtss_phy_status_1g_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_status_1g_t *const status)`  
*Get PHY 1G status.*
- `vtss_rc vtss_phy_power_conf_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_power_conf_t *const conf)`  
*Get PHY power configuration.*
- `vtss_rc vtss_phy_power_conf_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_power_conf_t *const conf)`  
*Set PHY power configuration.*
- `vtss_rc vtss_phy_power_status_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_power_status_t *const status)`  
*Get PHY power status.*
- `vtss_rc vtss_phy_clock_conf_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_recov_clk_t clock_port, const vtss_phy_clock_conf_t *const conf)`  
*Set PHY clock configuration.*

- `vtss_rc vtss_phy_clock_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_recov_clk_t` clock\_port, `vtss_phy_clock_conf_t` \*const conf, `vtss_port_no_t` \*const clock\_source)

*Get PHY clock configuration.*

- `vtss_rc vtss_phy_i2c_read` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `u8` i2c\_mux, const `u8` i2c\_reg\_addr, const `u8` i2c\_device\_addr, `u8` \*const value, `u8` cnt, `BOOL` word\_access)

*I2C read.*

- `vtss_rc vtss_phy_i2c_write` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `u8` i2c\_mux, const `u8` i2c\_reg\_addr, const `u8` i2c\_device\_addr, `u8` \*value, `u8` cnt, `BOOL` word\_access)

*I2C writes.*

- `vtss_rc vtss_phy_read` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `u32` addr, `u16` \*const value)

*Read value from PHY register.*

- `vtss_rc vtss_phy_read_page` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `u32` page, const `u32` addr, `u16` \*const value)

*Read value from PHY register at a specific page. Page register is set to standard page when read is done.*

- `vtss_rc vtss_phy_mmd_read` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `u16` devad, const `u32` addr, `u16` \*const value)

*Read value from PHY mmd register.*

- `vtss_rc vtss_phy_mmd_write` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `u16` devad, const `u32` addr, `u16` value)

*Write value to PHY mmd register.*

- `vtss_rc vtss_phy_write` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `u32` addr, const `u16` value)

*Write value to PHY register.*

- `vtss_rc vtss_phy_write_masked` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `u32` addr, const `u16` value, const `u16` mask)

*Write masked value to PHY register.*

- `vtss_rc vtss_phy_write_masked_page` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `u16` page, const `u16` addr, const `u16` value, const `u16` mask)

*Write masked value to PHY register and setups the page register.*

- `vtss_rc vtss_phy_gpio_mode` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `u8` gpio\_no, const `vtss_phy_gpio_mode_t` mode)

*Configure GPIO mode.*

- `vtss_rc vtss_phy_gpio_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `u8` gpio\_no, `BOOL` \*value)

*Get the value from a GPIO pin.*

- `vtss_rc vtss_phy_gpio_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `u8` gpio\_no, `BOOL` value)

*Set the value of a GPIO pin.*

- `vtss_rc vtss_phy_veriphy_start` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `u8` mode)

*Start VeriPHY.*

- `vtss_rc vtss_phy_veriphy_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_veriphy_result_t` \*const result)

*Get VeriPHY result.*

- `vtss_rc vtss_phy_led_mode_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_led_mode_select_t` led\_mode\_select)

*Setting the LEDs blink mode.*

- `vtss_rc vtss_phy_led_intensity_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_led_intensity` intensity)

*Setting the LEDs intensity.*

- `vtss_rc vtss_phy_led_intensity_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_led_intensity` \*intensity)

*Getting the LEDs intensity.*

- `vtss_rc vtss_phy_enhanced_led_control_init` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_enhanced_led_control_t` conf)
 

*Setting the enhanced LED control initial state (Should only be set once at startup)..*
- `vtss_rc vtss_phy_enhanced_led_control_init_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_enhanced_led_control_t` \*conf)
 

*Getting the enhanced LED control initial state (Should only be set once at startup)..*
- `vtss_rc vtss_phy_coma_mode_disable` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no)
 

*Pulling the coma mode pin low.*
- `vtss_rc vtss_phy_coma_mode_enable` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no)
 

*Pulling the coma mode pin high (Set in coma mode if no external hardware is controlling the pin)*
- `void vga_adc_debug` (const `vtss_inst_t` inst, `u8` vga\_adc\_pwr, `vtss_port_no_t` port\_no)
 

*debug function for Atom family Rev. A. chips*
- `vtss_rc vtss_phy_port_eee_capable` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `BOOL` \*eee\_capable)
 

*Get information about if a port is EEE capable.*
- `vtss_rc vtss_phy_eee_ena` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `BOOL` enable)
 

*Enabling / Disabling EEE (Energy Efficient Ethernet)*
- `vtss_rc vtss_phy_eee_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_eee_conf_t` \*conf)
 

*Getting the current EEE (Energy Efficient Ethernet) configuration.*
- `vtss_rc vtss_phy_eee_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_eee_conf_t` conf)
 

*Setting the EEE (Energy Efficient Ethernet) configuration.*
- `vtss_rc vtss_phy_eee_power_save_state_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `BOOL` \*rx\_in\_power\_save\_state, `BOOL` \*tx\_in\_power\_save\_state)
 

*Getting the if phy is currently powered save mode due to EEE.*
- `vtss_rc vtss_phy_eee_link_partner_advertisements_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `u8` \*advertisement)
 

*Getting the EEE advertisement.*
- `vtss_rc vtss_phy_event_enable_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_event_t` ev\_mask, const `BOOL` enable)
 

*Enabling / Disabling of events.*
- `vtss_rc vtss_phy_event_enable_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_event_t` \*ev\_mask)
 

*Getting current interrupt event state.*
- `vtss_rc vtss_phy_event_poll` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_event_t` \*const ev\_mask)
 

*Polling for active events.*
- `vtss_rc vtss_squelch_workaround` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `BOOL` enable)
 

*Function for enabling/disabling squelch work around.*
- `vtss_rc vtss_phy_csr_wr` (const `vtss_inst_t` inst, const `u16` page, const `vtss_port_no_t` port\_no, const `u16` target, const `u32` csr\_reg\_addr, const `u32` value)
 

*Function for writing to CSR registers.*
- `vtss_rc vtss_phy_csr_rd` (const `vtss_inst_t` inst, const `u16` page, const `vtss_port_no_t` port\_no, const `u16` target, const `u32` csr\_reg\_addr, `u32` \*value)
 

*Function for writing to CSR registers.*
- `vtss_rc vtss_phy_statistic_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_statistic_t` \*statistics)
 

*debug function for getting phy statistics.*
- `vtss_rc vtss_phy_do_page_chk_set` (const `vtss_inst_t` inst, const `BOOL` enable)
 

*Debug function for enabling check of page register for all phy register accesses.*
- `vtss_rc vtss_phy_do_page_chk_get` (const `vtss_inst_t` inst, `BOOL` \*enable)

- Debug function for getting if check of page register is enabled.*

  - `vtss_rc vtss_phy_loopback_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_loopback_t` loopback)

*Debug function for setting phy internal loopback.*

  - `vtss_rc vtss_phy_loopback_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_loopback_t` \*loopback)

*Debug function for getting the current phy internal loopback.*

  - `vtss_rc vtss_phy_is_8051_crc_ok` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `u16` code\_length, `u16` expected\_crc)

*Debug function for checking if the phy firmware is loaded correctly.*

  - `vtss_rc vtss_phy_cfg_ob_post0` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `u16` value)

*Debug function for setting the ob post0 patch.*

  - `vtss_rc vtss_phy_cfg_ib_cterm` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `u8` ib\_cterm\_ena, const `u8` ib\_eq\_mode)

*Debug function for setting the ib cterm patch.*

  - `vtss_rc vtss_phy_atom12_patch_settings_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `u8` \*mcu\_bus, `u8` \*cfg\_buf, `u8` \*stat\_buf)

*Debug function for getting PHY setting set by the micro patches.*

  - `void vtss_phy_reg_decode_status` (`vtss_port_no_t` port\_no, `u16` lp\_auto\_neg\_advertisement\_reg, `u16` lp←1000base\_t\_status\_reg, `u16` mii\_status\_reg, const `vtss_phy_conf_t` phy\_setup, `vtss_port_status_t` \*const status)

*Function for updating the status via the result from PHY registers.*

  - `vtss_rc vtss_phy_flowcontrol_decode_status` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `u16` lp←auto\_neg\_advertisement\_reg, const `vtss_phy_conf_t` phy\_setup, `vtss_port_status_t` \*const status)

*Function for finding flow control status based upon configuration and PHY registers.*

  - `vtss_rc vtss_phy_debug_stat_print` (const `vtss_inst_t` inst, const `vtss_debug_printf_t` pr, const `vtss_port_no_t` port\_no, const `BOOL` print\_hdr)

*debug function for printing PHY statistics*

  - `vtss_rc vtss_phy_warm_start_failed_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no)

*Function for checking if any issue were seen during warm-start.*

  - `vtss_rc vtss_phy_debug_phyinfo_print` (const `vtss_inst_t` inst, const `vtss_debug_printf_t` pr, const `vtss_port_no_t` port\_no, const `BOOL` print\_hdr)

*debug function for printing some of the internal PHY state/configurations*

  - `vtss_rc vtss_phy_debug_register_dump` (const `vtss_inst_t` inst, const `vtss_debug_printf_t` pr, `BOOL` clear, const `vtss_port_no_t` port\_no)

*debug function for printing some of the internal PHY state/configurations*

  - `vtss_rc vtss_phy_detect_base_ports` (const `vtss_inst_t` inst)

*Function for making getting the API updated with base port number for all ports. MUST not be called unless all ports are detected.*

  - `vtss_rc vtss_phy_ext_connector_loopback` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `BOOL` lpbk)

*Function for configuring External Connector Loopback.*

  - `vtss_rc vtss_phy_serdes_sgmii_loopback` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `u16` mode)

*Function for configuring MAC-SerDes(SGMII) Loopback.*

  - `vtss_rc vtss_phy_serdes_fmedia_loopback` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `u16` mode)

*Function for configuring Fibre-Media SerDes Loopback.*

  - `vtss_rc vtss_phy_debug_regdump_print` (const `vtss_inst_t` inst, const `vtss_debug_printf_t` pr, const `vtss_port_no_t` port\_no, const `vtss_port_no_t` page\_no, const `BOOL` print\_hdr)

*debug function for printing some of the internal PHY Registers*

  - `vtss_rc vtss_phy_wol_enable` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `BOOL` enable)

*function to Enable or Disable WOL by enabling or disabling the interrupt*

- `vtss_rc vtss_phy_wol_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_wol_conf_t` \*const conf)
 

*function to Get Wake-On-LAN configuration*
- `vtss_rc vtss_phy_wol_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_wol_conf_t` \*const conf)
 

*function to Set Wake-On-LAN configuration*
- `vtss_rc vtss_phy_patch_settings_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `u8` \*mcb\_bus, `u8` \*cfg\_buf, `u8` \*stat\_buf)
 

*Debug function for getting PHY setting set by the micro patches.*
- `vtss_rc vtss_phy_serdes6g_rcpll_status_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_rcpll_status_t` \*rcpll\_status)
 

*Debug function for getting PHY Serdes6G RC-PLL status.*
- `vtss_rc vtss_phy_serdes1g_rcpll_status_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_rcpll_status_t` \*rcpll\_status)
 

*Debug function for getting PHY Serdes1G RC-PLL status.*
- `vtss_rc vtss_phy_lcpll_status_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_lcpll_status_t` \*lcpll\_status)
 

*Debug function for getting PHY LC-PLL status.*
- `vtss_rc vtss_phy_reset_lcpll` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no)
 

*Debug function for Resetting the LCPLL for the PHY.*
- `vtss_rc vtss_phy_sd6g_ob_post_rd` (`vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `u8` \*ob\_post0, `u8` \*ob\_post1)
 

*Debug function for reading the 6G SerDes ob\_post0 and ob\_post1 values.*
- `vtss_rc vtss_phy_sd6g_ob_post_wr` (`vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `u8` ob\_post0, const `u8` ob\_post1)
 

*Debug function for modifying the 6G SerDes ob\_post0 and ob\_post1 values.*
- `vtss_rc vtss_phy_sd6g_ob_lev_rd` (`vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `u8` \*ob\_level)
 

*Debug function for reading the 6G SerDes ob\_lev value.*
- `vtss_rc vtss_phy_sd6g_ob_lev_wr` (`vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `u8` ob\_level)
 

*Debug function for modifying the 6G SerDes ob\_lev value.*
- `vtss_rc vtss_phy_mac_media_inhibit_odd_start` (`vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `BOOL` mac\_inhibit, const `BOOL` media\_inhibit)
 

*Function to modify the values for the MAC and MEDIA I/F Inhibit ODD Start Delay.*
- `vtss_rc vtss_phy_fefi_get` (`vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_fefi_mode_t` \*fefi)
 

*Function to modify the values for the Far-End Fail Indication.*
- `vtss_rc vtss_phy_fefi_set` (`vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_fefi_mode_t` fefi)
 

*Function to modify the values for the Far-End Fail Indication.*
- `vtss_rc vtss_phy_fefi_detect` (`vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `BOOL` \*fefi\_detect)
 

*Function to get the status for the Far-End Fail Indication.*
- `vtss_rc vtss_phy_mse_100m_get` (`vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `u32` \*mse)
 

*Function to get the Mean Squared Error (MSE), ie. Noise floor for 100m.*
- `vtss_rc vtss_phy_mse_1000m_get` (`vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `u32` \*mseA, `u32` \*mseB, `u32` \*mseC, `u32` \*mseD)
 

*Function to get the Mean Squared Error (MSE), ie. Noise floor for 1000m.*
- `vtss_rc vtss_phy_read_tr_addr` (`vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `u16` tr\_addr, `u16` \*tr\_lower, `u16` \*tr\_upper)
 

*Debug Function to retrieve the values from Token Ring.*
- `vtss_rc vtss_phy_is_viper_revB` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `BOOL` \*is\_viper\_revB)
 

*Polling for to determine if the Chip Type and revision is Viper Rev\_B.*
- `vtss_rc vtss_phy_ext_event_poll` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_event_t` \*const ev\_mask)
 

*Polling for active EXT Interrupt events.*

- `vtss_rc vtss_phy_status_inst_poll` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_port_status_t` \*const status)  
*Get PHY status from the PHY Instance (Does not read PHY Registers).*
- `vtss_rc vtss_phy_macsec_csr_sd6g_rd` (`vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `u16` target, const `u32` csr\_reg\_addr, `u32` \*value)  
*Function for Reading from 6G serdes registers (applicable only on VIPER,ELISE)*
- `vtss_rc vtss_phy_macsec_csr_sd6g_wr` (`vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `u16` target, const `u32` csr\_reg\_addr, `u32` value)  
*Function for Writing to 6G serdes registers (applicable only on VIPER,ELISE)*

### 6.31.1 Detailed Description

PHY API.

This header file describes PHY control functions

### 6.31.2 Macro Definition Documentation

#### 6.31.2.1 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.

#### 6.31.2.2 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.

#### 6.31.2.3 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.

### 6.31.2.4 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.

### 6.31.2.5 VTSS\_PHY\_ACTIPHY\_PWR

```
#define VTSS_PHY_ACTIPHY_PWR 100
```

ActiPHY power status

Definition at line 592 of file vtss\_phy\_api.h.

### 6.31.2.6 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.

### 6.31.2.7 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.

### 6.31.2.8 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.

### 6.31.2.9 VTSS\_PHY\_RECov\_CLK2

```
#define VTSS_PHY_RECov_CLK2 1
```

RCVRD\_CLK2

Definition at line 618 of file vtss\_phy\_api.h.

### 6.31.2.10 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.

### 6.31.2.11 VTSS\_PHY\_PAGE\_STANDARD

```
#define VTSS_PHY_PAGE_STANDARD 0x0000
```

Standard registers

Definition at line 740 of file vtss\_phy\_api.h.

### 6.31.2.12 VTSS\_PHY\_PAGE\_EXTENDED

```
#define VTSS_PHY_PAGE_EXTENDED 0x0001
```

Extended registers

Definition at line 741 of file vtss\_phy\_api.h.

### 6.31.2.13 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.

### 6.31.2.14 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.

### 6.31.2.15 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.

### 6.31.2.16 VTSS\_PHY\_PAGE\_GPIO

```
#define VTSS_PHY_PAGE_GPIO 0x0010
```

GPIO registers

Definition at line 745 of file vtss\_phy\_api.h.

### 6.31.2.17 VTSS\_PHY\_PAGE\_1588

```
#define VTSS_PHY_PAGE_1588 0x1588
```

1588 (PTP) registers

Definition at line 746 of file vtss\_phy\_api.h.

### 6.31.2.18 VTSS\_PHY\_PAGE\_MACSEC

```
#define VTSS_PHY_PAGE_MACSEC 0x0004
```

MACSEC page

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

### 6.31.2.19 VTSS\_PHY\_PAGE\_TEST

```
#define VTSS_PHY_PAGE_TEST 0x2A30
```

Test registers

Definition at line 748 of file vtss\_phy\_api.h.

### 6.31.2.20 VTSS\_PHY\_PAGE\_TR

```
#define VTSS_PHY_PAGE_TR 0x52B5
```

Token ring registers

Definition at line 749 of file vtss\_phy\_api.h.

### 6.31.2.21 VTSS\_PHY\_PAGE\_0x2DAF

```
#define VTSS_PHY_PAGE_0x2DAF 0x2DAF
```

0x2DAF registers

Definition at line 750 of file vtss\_phy\_api.h.

### 6.31.2.22 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.

### 6.31.2.23 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.

### 6.31.2.24 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.

### 6.31.2.25 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.

### 6.31.2.26 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.

### 6.31.2.27 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.

### 6.31.2.28 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.

### 6.31.2.29 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.

### 6.31.2.30 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.

### 6.31.2.31 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.

### 6.31.2.32 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.

### 6.31.2.33 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.

**6.31.2.34 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.

**6.31.2.35 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.

**6.31.2.36 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.

**6.31.2.37 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.

**6.31.2.38 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.

**6.31.2.39 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.

**6.31.2.40 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.

**6.31.2.41 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.

**6.31.2.42 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.

**6.31.2.43 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.

**6.31.2.44 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.

**6.31.2.45 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.

**6.31.2.46 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.

**6.31.2.47 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.

**6.31.2.48 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.

**6.31.2.49 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.

**6.31.2.50 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.

**6.31.2.51 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.

**6.31.2.52 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.

**6.31.2.53 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.

**6.31.2.54 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.

**6.31.2.55 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.

**6.31.2.56 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.

**6.31.2.57 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.

**6.31.3 Typedef Documentation****6.31.3.1 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.

### 6.31.3.2 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.

## 6.31.4 Enumeration Type Documentation

### 6.31.4.1 vtss\_phy\_led\_mode\_t

```
enum vtss_phy_led_mode_t
```

PHY LED modes.

#### Enumerator

LINK1000_ACTIVITY	No link in any speed on any media interface./Valid link at any speed on any media interface. Blink or pulse-stretch = Valid link at any speed on any media interface with activity present.
LINK100_ACTIVITY	No link in 1000BASE-T/Valid 1000BASE-T link. Blink or pulse-stretch = Valid 1000BASE-T link with activity present
LINK10_ACTIVITY	No link in 100BASE-T/Valid 100BASE-T link. Blink or pulse-stretch = Valid 100BASE-T link with activity present
LINK100_1000_ACTIVITY	No link in 10BASE-T/Valid 10BASE-T link. Blink or pulse-stretch = Valid 10BASE-T link with activity present
LINK10_1000_ACTIVITY	No link in 100BASE-T, 1000BASE-FX, or 1000BASE-TX/Valid 100BASE-T, 1000BASE-FX, or 1000BASE-TX link.Blink or pulse-stretch = Valid 100BASE-T, 1000BASE-FX, or 1000BASE-TX link with activity present.
LINK10_100_ACTIVITY	No link in 10BASE-T, 1000BASE-FX, or 1000BASE-TX/Valid 10BASE-T, 1000BASE-FX, or 1000BASE-TX link.Blink or pulse-stretch = Valid 10BASE-T, 1000BASE-FX, or 1000BASE-TX link with activity present.
LINK100BASE_FX_1000BASE_X_ACTIVITY	No link in 10BASE-T, 100BASE-FX, or 100BASE-TX/Valid 10BASE-T, 100BASE-FX, or 100BASE-TX link.Blink or pulse-stretch = Valid 10BASE-T, 100BASE-FX, or 100BASE-TX link with activity present.
DUPLEX_COLLISION	No link in 100BASE-FX or 1000BASE-X/ Valid 100BASE-FX or 1000BASE-X link. Blink or pulse-stretch = Valid 100BASE-FX or 1000BASE-X link with activity present.
COLLISION	Link established in half-duplex mode, or no link established. Link established in full-duplex mode.Blink or pulse-stretch = Link established in half-duplex mode but collisions are present

## Enumerator

ACTIVITY	No collision detected.Blink or pulse-stretch = Collision detected.
BASE100_FX_1000BASE_X_FIBER_ACTIVITY	No activity present.Blink or pulse-stretch = Activity present
AUTONEGOTIATION_FAULT	No 100BASE-FX or 1000BASE-X activity present. Blink or pulse-stretch = 100BASE-FX or 1000BASE-X activity present
LINK1000BASE_X_ACTIVITY	No autonegotiation fault present., Autonegotiation fault occurred.
LINK100BASE_FX_ACTIVITY	No link in 1000BASE-X. Valid 1000BASE-X link.
BASE1000_ACTIVITY	No link in 100BASE-FX.
BASE100_FX_ACTIVITY	No 1000BASE-X activity present.Blink or pulse-stretch = 1000BASE-X activity present.
FORCE_LED_OFF	No 100BASE-FX activity present, Blink or pulse-stretch = 100BASE-FX activity present.
FORCE_LED_ON	De-asserts the LED
FAST_LINK_FAIL	Asserts the LED

Definition at line 102 of file vtss\_phy\_api.h.

## 6.31.4.2 vtss\_phy\_media\_interface\_t

```
enum vtss_phy_media_interface_t
```

PHY media interface type.

## Enumerator

VTSS_PHY_MEDIA_IF_CU	Copper Interface
VTSS_PHY_MEDIA_IF_SFP_PASSTHRU	Fiber/Cu SFP Pass-thru
VTSS_PHY_MEDIA_IF_FI_1000BX	1000Base-X
VTSS_PHY_MEDIA_IF_FI_100FX	100Base-FX
VTSS_PHY_MEDIA_IF_AMS CU PASSTHRU	AMS - Cat5/SerDes/CuSFP passthru - Note the phy mode must be set to VTSS_PHY_MODE_ANEG
VTSS_PHY_MEDIA_IF_AMS FI PASSTHRU	AMS - Fiber passthru - Note the phy mode must be set to VTSS_PHY_MODE_ANEG
VTSS_PHY_MEDIA_IF_AMS CU_1000BX	AMS - Cat5/1000BX/CuSFP
VTSS_PHY_MEDIA_IF_AMS CU_100FX	AMS - Cat5/100FX/CuSFP

Definition at line 161 of file vtss\_phy\_api.h.

## 6.31.4.3 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.

## 6.31.4.4 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.

## 6.31.4.5 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.

#### 6.31.4.6 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.

#### 6.31.4.7 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.

#### 6.31.4.8 vtss\_phy\_fast\_link\_fail\_t

enum [vtss\\_phy\\_fast\\_link\\_fail\\_t](#)

PHY fast link failure pin enable/disable.

Enumerator

VTSS_PHY_FAST_LINK_FAIL_ENABLE	Enable fast link failure pin
VTSS_PHY_FAST_LINK_FAIL_DISABLE	Disable fast link failure pin

Definition at line 325 of file vtss\_phy\_api.h.

#### 6.31.4.9 `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`.

#### 6.31.4.10 `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`.

#### 6.31.4.11 `vtss_phy_mac_serdes_pcs_sgmii_pre`

enum `vtss_phy_mac_serdes_pcs_sgmii_pre`

PHY MEDIA SerDes PCS Remote Fault Indication, See Clause 37, Table 37-3.

Enumerator

<code>VTSS_PHY_MAC_SERD_PCS_SGMII_IN_PRE_</code> ↔ NONE	MAC SerDes PCS Control, SGMII Input Preamble for 100BaseX - No Preamble Req'd
<code>VTSS_PHY_MAC_SERD_PCS_SGMII_IN_PRE_</code> ↔ ONE	MAC SerDes PCS Control, SGMII Input Preamble for 100BaseX - One-Byte Preamble Req'd
<code>VTSS_PHY_MAC_SERD_PCS_SGMII_IN_PRE_</code> ↔ TWO	MAC SerDes PCS Control, SGMII Input Preamble for 100BaseX - Two-Byte Preamble Req'd
<code>VTSS_PHY_MAC_SERD_PCS_SGMII_IN_PRE_</code> ↔ RSVD	MAC SerDes PCS Control, SGMII Input Preamble for 100BaseX - Reserved

Definition at line 343 of file `vtss_phy_api.h`.

## 6.31.4.12 vtss\_phy\_media\_rem\_fault\_t

```
enum vtss_phy_media_rem_fault_t
```

PHY MEDIA SerDes PCS Remote Fault Indication, See Clause 37, Table 37-3.

## Enumerator

VTSS_PHY_MEDIA_SERD_PCS_Rem_Fault_No_Error	Media SerDes PCS Control, Most Recent Clause 37 ANEG Exchg - Table 37-3
VTSS_PHY_MEDIA_SERD_PCS_Rem_Fault_Offline	Media SerDes PCS Control, Most Recent Clause 37 ANEG Exchg
VTSS_PHY_MEDIA_SERD_PCS_Rem_Fault_LINK_Fail	Media SerDes PCS Control, Most Recent Clause 37 ANEG Exchg
VTSS_PHY_MEDIA_SERD_PCS_Rem_Fault_Auto_Neg_Error	Media SerDes PCS Control, Most Recent Clause 37 ANEG Exchg

Definition at line 367 of file vtss\_phy\_api.h.

## 6.31.4.13 vtss\_phy\_media\_force\_ams\_sel\_t

```
enum vtss_phy_media_force_ams_sel_t
```

PHY AMS Force configuration.

## Enumerator

VTSS_PHY_MEDIA_FORCE_AMS_Selection_Normal	Force AMS Override to Force Selection - Normal
VTSS_PHY_MEDIA_FORCE_AMS_Selection_Serdes	Force AMS Override to Force Selection - SerDes Media
VTSS_PHY_MEDIA_FORCE_AMS_Selection_Copper	Force AMS Override to Force Selection - Copper Media

Definition at line 388 of file vtss\_phy\_api.h.

## 6.31.4.14 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.

#### 6.31.4.15 vtss\_phy\_freq\_t

enum `vtss_phy_freq_t`

PHY clock frequencies.

Enumerator

<code>VTSS_PHY_FREQ_25M</code>	25 MHz
<code>VTSS_PHY_FREQ_125M</code>	125 MHz
<code>VTSS_PHY_FREQ_3125M</code>	31.25 MHz This is only valid on ATOM family - NOT Enzo

Definition at line 632 of file vtss\_phy\_api.h.

#### 6.31.4.16 vtss\_phy\_clk\_squelch

enum `vtss_phy_clk_squelch`

PHY clock squelch levels.

Enumerator

<code>VTSS_PHY_CLK_SQUELCH_MAX</code>	Automatically squelch clock to low when the link is not up, is unstable, is up in a mode that does not support the generation of arecovered clock (1000BASE-T master or 10BASE-T), or is up in EEE mode (100BASE-TX or 1000BASE-T slave).
<code>VTSS_PHY_CLK_SQUELCH_MED</code>	Same as <code>VTSS_PHY_CLK_SQUELCH_MAX</code> except that the clock is also generated in 1000BASE-T master and 10BASE-T link-up modes. This mode also generates a recovered clock output in EEE mode during reception of LP_IDLE.
<code>VTSS_PHY_CLK_SQUELCH_MIN</code>	Squelch only when the link is not up
<code>VTSS_PHY_CLK_SQUELCH_NONE</code>	Disable clock squelch.

Definition at line 639 of file vtss\_phy\_api.h.

#### 6.31.4.17 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.

**6.31.4.18 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.

**6.31.4.19 vtss\_wol\_passwd\_len\_type\_t**

```
enum vtss_wol_passwd_len_type_t
```

Structure for Wake-On-LAN Password Length configuration.

**Enumerator**

VTSS_WOL_PASSWD_LEN←_4	PasswdLen=4 bytes
VTSS_WOL_PASSWD_LEN←_6	PasswdLen=6 bytes

Definition at line 1672 of file vtss\_phy\_api.h.

### 6.31.4.20 vtss\_fefi\_mode\_t

enum `vtss_fefi_mode_t`

Far-End Failure Indication modes These settings map to PHY Reg23E3, bits 0 and 1.

#### Enumerator

<code>VTSS_100FX_FEFI_NORMAL</code>	Normal FEFI Operation, as specified by Reg23E3.1
<code>VTSS_100FX_FEFI_FORCE_SUPPRESS</code>	Force FEFI, as specified by Reg23E3.0
<code>VTSS_100FX_FEFI_FORCE_ENABLE</code>	Force FEFI, as specified by Reg23E3.0

Definition at line 1900 of file `vtss_phy_api.h`.

## 6.31.5 Function Documentation

### 6.31.5.1 vtss\_phy\_pre\_reset()

```
vtss_rc vtss_phy_pre_reset (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no )
```

Must be call previous to port PHY Reset (`vtss_phy_reset`).

#### Parameters

<code>inst</code>	[IN] Target instance reference.
<code>port_no</code>	[IN] Port number (MUST be the first port for the chip).

#### Returns

Return code.

### 6.31.5.2 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.

**6.31.5.3 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.

**6.31.5.4 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.

### 6.31.5.5 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.

### 6.31.5.6 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.

### 6.31.5.7 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.

**6.31.5.8 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.

**6.31.5.9 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.

**6.31.5.10 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.

**6.31.5.11 vtss\_phy\_cl37\_lp\_abil\_get()**

```
vtss_rc vtss_phy_cl37_lp_abil_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_port_status_t *const status )
```

Get Clause37 Link pArtner's ability.

**Parameters**

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>status</i>	[OUT] PHY status.

**Returns**

Return code.

### 6.31.5.12 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.

### 6.31.5.13 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.

### 6.31.5.14 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.

**6.31.5.15 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.

**6.31.5.16 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.

**6.31.5.17 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.

**6.31.5.18 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.

### 6.31.5.19 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.

### 6.31.5.20 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.

### 6.31.5.21 vtss\_phy\_i2c\_read()

```
vtss_rc vtss_phy_i2c_read (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const u8 i2c_mux,
    const u8 i2c_reg_addr,
    const u8 i2c_device_addr,
    u8 *const value,
    u8 cnt,
    BOOL word_access )
```

I2C read.

#### Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>i2c_mux</i>	[IN] The i2c clock mux
<i>i2c_reg_addr</i>	[IN] The i2c register address to access.
<i>i2c_device_addr</i>	[IN] The i2c address of the device to access.
<i>value</i>	[OUT] Pointer to where array which in going to contain the values read.
<i>cnt</i>	[IN] The number of registers to read. Note: The reg_addr is incremented by 1 for each of the read counts. If you want to read 16 bites registers (2 times 8 bits from the same register address), you need to do that by calling the vtss_phy_i2c_read twice, and not use the cnt (set cnt to 1).
<i>word_access</i>	[IN] Set to TRUE if the register data width is 16bit. FALSE = 8 bits data width

#### Returns

Return code.

### 6.31.5.22 vtss\_phy\_i2c\_write()

```
vtss_rc vtss_phy_i2c_write (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const u8 i2c_mux,
    const u8 i2c_reg_addr,
    const u8 i2c_device_addr,
    u8 * value,
    u8 cnt,
    BOOL word_access )
```

I2C writes.

#### Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>i2c_mux</i>	[IN] The i2c clock mux

**Parameters**

<i>i2c_reg_addr</i>	[IN] The i2c register address to access.
<i>i2c_device_addr</i>	[IN] The i2c address of the device to access.
<i>value</i>	[IN] Pointer to where array containing the values to write.
<i>cnt</i>	[IN] The number of registers to write. Note: The reg_addr is incremented by 1 for each of the write counts. If you want to write 16 bites registers (2 times 8 bits to the same register address), you need to do that by calling the vtss_phy_i2c_write twice, and not use the cnt (set cnt to 1).
<i>word_access</i>	[IN] Set to TRUE if the register data width is 16bit. FALSE = 8 bits data width

**Returns**

Return code.

**6.31.5.23 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.

**6.31.5.24 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.

**6.31.5.25 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.

**6.31.5.26 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.

**6.31.5.27 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.

**6.31.5.28 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.

**6.31.5.29 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.

**6.31.5.30 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.

**6.31.5.31 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.

**6.31.5.32 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.

**6.31.5.33 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.

**6.31.5.34 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.

### 6.31.5.35 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.

### 6.31.5.36 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.

### 6.31.5.37 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.

**6.31.5.38 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.

**6.31.5.39 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.

**6.31.5.40 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.

**6.31.5.41 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.

**6.31.5.42 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).

### 6.31.5.43 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.

### 6.31.5.44 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.

**6.31.5.45 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.

**6.31.5.46 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.

### 6.31.5.47 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.

### 6.31.5.48 vtss\_phy\_eee\_link\_partner\_advertisements\_get()

```
vtss_rc vtss_phy_eee_link_partner_advertisements_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    u8 * advertisement )
```

Getting the EEE advertisement.

#### Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number
<i>advertisement</i>	[OUT] Advertisement bit mask. Bit 0 = Link partner advertises 100BASE-T capability. Bit 1 = Link partner advertises 1000BASE-T capability.

#### Returns

Return code.

### 6.31.5.49 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.

### 6.31.5.50 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.

### 6.31.5.51 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.

**6.31.5.52 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

**6.31.5.53 vtss\_phy\_csr\_wr()**

```
vtss_rc vtss_phy_csr_wr (
    const vtss_inst_t inst,
    const u16 page,
    const vtss_port_no_t port_no,
    const u16 target,
    const u32 csr_reg_addr,
    const u32 value )
```

Function for writing to CSR registers.

**Parameters**

<i>inst</i>	[IN] Target instance reference.
-------------	---------------------------------

**Parameters**

<i>port_no</i>	[IN] The port in question
<i>page</i>	[IN] The page for the CSR register access e.g. VTSS_PHY_PAGE_1588 or VTSS_PHY_PAGE_MACSEC
<i>target</i>	[IN] The CSR target
<i>csr_reg_addr</i>	[IN] The CSR register to write
<i>value</i>	[IN] The value to write

**Returns**

VTSS\_RC\_OK when new access can be done - VTSS\_RC\_ERROR if something went wrong and access was never granted.

**6.31.5.54 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.

**6.31.5.55 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

### 6.31.5.56 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.

### 6.31.5.57 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.

**6.31.5.58 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

**6.31.5.59 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

### 6.31.5.60 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

### 6.31.5.61 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

### 6.31.5.62 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

### 6.31.5.63 vtss\_phy\_atom12\_patch\_settings\_get()

```
vtss_rc vtss_phy_atom12_patch_settings_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    u8 * mcb_bus,
    u8 * cfg_buf,
    u8 * stat_buf )
```

Debug function for getting PHY setting set by the micro patches.

#### Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] PHY port within the chip.
<i>mcb_bus</i>	[INOUT] If set to 2 the returned data is for LCPLL/RComp else the mcb_bus is returning 0 if the data is for 1G, and returning 1 if the data is for 6G .
<i>cfg_buf</i>	[IN] Pointer to array where to put the current settings.
<i>stat_buf</i>	[IN] Pointer to array where to put the current status.

#### Returns

Return code. VTSS\_RC\_OK if patch were updated correct else VTSS\_RC\_ERROR

### 6.31.5.64 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

#### 6.31.5.65 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 no errors were seen during warm-start else VTSS\_RC\_ERROR.

#### 6.31.5.66 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.

### 6.31.5.67 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.

### 6.31.5.68 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.

**6.31.5.69 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.

**6.31.5.70 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.

**6.31.5.71 vtss\_phy\_ext\_connector\_loopback()**

```
vtss_rc vtss_phy_ext_connector_loopback (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const BOOL lpback )
```

Function for configuring External Connector Loopback.

**Parameters**

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] The port in question.
<i>lpback</i>	[IN] TRUE=Loopback ON, FALSE=Loopback OFF

**Returns**

Return code. VTSS\_RC\_OK if no errors were seen during warm-start else VTSS\_RC\_ERROR.

**6.31.5.72 vtss\_phy\_serdes\_sgmii\_loopback()**

```
vtss_rc vtss_phy_serdes_sgmii_loopback (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const u16 mode )
```

Function for configuring MAC-SerDes(SGMII) Loopback.

**Parameters**

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] The port in question.
<i>mode</i>	[IN] serdes mode and port mode: bits 15:12: 0x8= Fibre Media or 0x9=SGMII/QSGMII MAC mode: bits 11:8: Port Address (0-3) mode: bits 7:4: Loopback Type: 0=No,1=Pad,2=Input,4=Facility,8=Equipment mode: bits 3:0: 0x2

**Returns**

Return code. VTSS\_RC\_OK if no errors were seen during warm-start else VTSS\_RC\_ERROR.

### 6.31.5.73 vtss\_phy\_serdes\_fmedia\_loopback()

```
vtss_rc vtss_phy_serdes_fmedia_loopback (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const u16 mode )
```

Function for configuring Fibre-Media SerDes Loopback.

#### Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] The port in question.
<i>mode</i>	[IN] serdes mode and port mode: bits 15:12: 0x8= Fibre Media or 0x9=SGMII/QSGMII MAC mode: bits 11:8: Port Address (0-3) mode: bits 7:4: Loopback Type: 0=No,1=Pad,2=Input,4=Facility,8=Equipment mode: bits 3:0: 0x2

#### Returns

Return code. VTSS\_RC\_OK if not errors were seen during warm-start else VTSS\_RC\_ERROR.

### 6.31.5.74 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.

### 6.31.5.75 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.

### 6.31.5.76 vtss\_phy\_wol\_conf\_get()

```
vtss_rc vtss_phy_wol_conf_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_wol_conf_t *const conf )
```

function to Get Wake-On-LAN configuration

#### Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port in question
<i>conf</i>	[IN] Ptr to WoL Structure <code>vtss_phy_wol_conf_t</code> to be filled out by API

#### Returns

Return code. VTSS\_RC\_OK if no errors.

### 6.31.5.77 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.

**6.31.5.78 vtss\_phy\_patch\_settings\_get()**

```
vtss_rc vtss_phy_patch_settings_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    u8 * mcb_bus,
    u8 * cfg_buf,
    u8 * stat_buf )
```

Debug function for getting PHY setting set by the micro patches.

**Parameters**

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] PHY port within the chip.
<i>mcb_bus</i>	[INOUT] If set to 2 the returned data is for LCPLL/RComp else the mcb_bus is returning 0 if the data is for 1G, and returning 1 if the data is for 6G .
<i>cfg_buf</i>	[IN] Pointer to array where to put the current settings.
<i>stat_buf</i>	[IN] Pointer to array where to put the current status.

**Returns**

Return code. VTSS\_RC\_OK if patch were updated correct else VTSS\_RC\_ERROR

**6.31.5.79 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

**6.31.5.80 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

**6.31.5.81 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

**6.31.5.82 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.

**6.31.5.83 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.

**6.31.5.84 vtss\_phy\_sd6g\_ob\_post\_wr()**

```
vtss_rc vtss_phy_sd6g_ob_post_wr (
    vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const u8 ob_post0,
    const u8 ob_post1 )
```

Debug function for modifying the 6G SerDes ob\_post0 and ob\_post1 values.

**Parameters**

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] PHY port within the chip.
<i>ob_post0</i>	[IN] Modify ob_post0 settings in serdes6g_ob_cfg for 6G SerDes Macro (See TN1052), 6 bits, default:0, Customizeable.
<i>ob_post1</i>	[IN] Modify ob_post1 settings in serdes6g_ob_cfg for 6G SerDes Macro (See TN1052), 5 bits, default:0, Do not change.

**Returns**

Return code. VTSS\_RC\_OK if all Ok VTSS\_RC\_ERROR if and error occurred.

**6.31.5.85 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.

### 6.31.5.86 vtss\_phy\_sd6g\_ob\_lev\_wr()

```
vtss_rc vtss_phy_sd6g_ob_lev_wr (
    vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const u8 ob_level )
```

Debug function for modifying the 6G SerDes ob\_lev value.

#### Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] PHY port within the chip.
<i>ob_level</i>	[IN] Modify ob_lev settings in serdes6g_ob_cfg1 for 6G SerDes Macro (See TN1052), 6 bits, default:0x18, Customizable Amplitude Control.

#### Returns

Return code. VTSS\_RC\_OK if all Ok VTSS\_RC\_ERROR if and error occurred.

### 6.31.5.87 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.

### 6.31.5.88 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.

### 6.31.5.89 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.

### 6.31.5.90 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.

**6.31.5.91 vtss\_phy\_mse\_100m\_get()**

```
vtss_rc vtss_phy_mse_100m_get (
    vtss_inst_t inst,
    const vtss_port_no_t port_no,
    u32 * mse )
```

Function to get the Mean Squared Error (MSE), ie. Noise floor for 100m.

**Parameters**

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] PHY port within the chip.
<i>mse</i>	[OUT] PHY port MSE Value In order to convert the returned mse value to dB, do the following This computation would be required for each channel pair $mse\_dbl = mse / (1024 * 2048)$ ; Note: Convert to Double and Scale by $2^{21} = 1024 * 2048$ $mse\_dbl = 20 * \log10(mse\_dbl)$ ; Note: Convert to dB Nominal Computed Values for $mse\_dbl$ are: -22dB to -31dB

**Returns**

Return code. VTSS\_RC\_OK if all Ok VTSS\_RC\_ERROR if and error occurred.

**6.31.5.92 vtss\_phy\_mse\_1000m\_get()**

```
vtss_rc vtss_phy_mse_1000m_get (
    vtss_inst_t inst,
    const vtss_port_no_t port_no,
    u32 * mseA,
    u32 * mseB,
    u32 * mseC,
    u32 * mseD )
```

Function to get the Mean Squared Error (MSE), ie. Noise floor for 1000m.

**Parameters**

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] PHY port within the chip.
<i>mseA</i>	[OUT] PHY port MSE Chan A Value
<i>mseB</i>	[OUT] PHY port MSE Chan B Value
<i>mseC</i>	[OUT] PHY port MSE Chan C Value
<i>mseD</i>	[OUT] PHY port MSE Chan D Value In order to convert the returned mse value to dB, do the following This computation would be required for each channel pair $mse\_dbl = mse / (1024 * 2048)$ ; Note: Convert to Double and Scale by $2^{21} = 1024 * 2048$ $mse\_dbl = 20 * \log_{10}(mse\_dbl)$ ; Note: Convert to dB Nominal Computed Values for $mse\_dbl$ are: -22dB to -31dB

**Returns**

Return code. VTSS\_RC\_OK if all Ok VTSS\_RC\_ERROR if and error occurred.

**6.31.5.93 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.

**6.31.5.94 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.

**6.31.5.95 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.

**6.31.5.96 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.

**6.31.5.97 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.

**6.31.5.98 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.

## 6.32 vtss\_api/include/vtss\_phy\_ts\_api.h File Reference

PHY TimeStamping API.

```
#include <vtss/api/types.h>
#include <vtss/api/port.h>
#include "vtss_misc_api.h"
```

### Data Structures

- struct [vtss\\_phy\\_ts\\_alt\\_clock\\_mode\\_s](#)  
*parameter for setting the alternative clock mode.*
- struct [vtss\\_phy\\_ts\\_pps\\_config\\_s](#)  
*PPS Configuration.*
- struct [vtss\\_phy\\_timestamp\\_t](#)  
*PHY timestamp in seconds and nanoseconds (10 bytes Timestamp)*
- struct [vtss\\_phy\\_ts\\_sertod\\_conf\\_t](#)  
*PHY timestamp in seconds and nanoseconds (10 bytes Timestamp)*
- struct [vtss\\_phy\\_ltc\\_freq\\_synth\\_s](#)  
*Frequency synthesis pulse configuration.*
- struct [vtss\\_phy\\_daisy\\_chain\\_conf\\_t](#)  
*SPI daisy chain configuration.*
- struct [vtss\\_phy\\_ts\\_fifo\\_sig\\_t](#)  
*Tx TSFIFO entry signature.*
- struct [vtss\\_phy\\_ts\\_eng\\_init\\_conf\\_t](#)  
*Defines the basic engine parameters passed to the engine\_init\_conf\_get() function.*
- struct [vtss\\_phy\\_ts\\_eth\\_conf\\_t](#)  
*Analyzer Ethernet comparator configuration options.*
- struct [vtss\\_phy\\_ts\\_ip\\_conf\\_t](#)  
*Analyzer IP comparator configuration options.*
- struct [vtss\\_phy\\_ts\\_mpls\\_lvl\\_rng\\_t](#)  
*MPLS level range.*
- struct [vtss\\_phy\\_ts\\_mpls\\_conf\\_t](#)  
*Analyzer MPLS comparator configuration options.*
- struct [vtss\\_phy\\_ts\\_ach\\_conf\\_t](#)  
*Analyzer ACH comparator configuration options.*
- struct [vtss\\_phy\\_ts\\_gen\\_conf\\_t](#)  
*Analyzer Generic data configuration options using IP comparator.*
- struct [vtss\\_phy\\_ts\\_ptp\\_engine\\_flow\\_conf\\_t](#)  
*PTP engine flow configuration options.*
- struct [vtss\\_phy\\_ts\\_oam\\_engine\\_flow\\_conf\\_t](#)  
*OAM engine flow configuration options.*
- struct [vtss\\_phy\\_ts\\_generic\\_flow\\_conf\\_t](#)  
*Generic engine flow configuration options.*

- struct [vtss\\_phy\\_ts\\_engine\\_flow\\_conf\\_t](#)  
*Analyzer flow configuration options.*
- struct [vtss\\_phy\\_ts\\_ptp\\_conf\\_t](#)  
*Analyzer PTP comparator configuration options.*
- struct [vtss\\_phy\\_ts\\_ptp\\_engine\\_action\\_t](#)  
*Analyzer PTP action configuration options.*
- struct [vtss\\_phy\\_ts\\_y1731\\_oam\\_conf\\_t](#)  
*Analyzer OAM comparator, Y.1731 OAM Packet format configuration options.*
- struct [vtss\\_phy\\_ts\\_ietf\\_mpls\\_ach\\_oam\\_conf\\_t](#)  
*Analyzer OAM comparator, IETF MPLS ACH OAM Packet format configuration options.*
- struct [vtss\\_phy\\_ts\\_oam\\_engine\\_action\\_t](#)  
*OAM Action configuration options.*
- struct [vtss\\_phy\\_ts\\_generic\\_action\\_t](#)  
*Generic Action configuration option.*
- struct [vtss\\_phy\\_ts\\_engine\\_action\\_t](#)  
*Engine Action configuration options.*
- struct [vtss\\_phy\\_ts\\_stats\\_t](#)  
*Timestamping Statistics.*
- struct [vtss\\_phy\\_ts\\_init\\_conf\\_t](#)  
*Defines the initial parameters to be passed to init function.*
- struct [vtss\\_phy\\_ts\\_nphase\\_status\\_t](#)  
*n-phase status*
- struct [vtss\\_phy\\_10g\\_fifo\\_sync\\_t](#)  
*10G OOS workaround options*
- struct [vtss\\_phy\\_ts\\_fifo\\_conf\\_t](#)  
*Defines the params for FIFO SYNC function.*

## Macros

- #define [VTSS\\_PHY\\_TS\\_FIFO\\_SIG\\_SRC\\_IP](#) 0x01  
*Defines Tx TSFIFO signature mask.*
- #define [VTSS\\_PHY\\_TS\\_FIFO\\_SIG\\_DEST\\_IP](#) 0x02
- #define [VTSS\\_PHY\\_TS\\_FIFO\\_SIG\\_MSG\\_TYPE](#) 0x04
- #define [VTSS\\_PHY\\_TS\\_FIFO\\_SIG\\_DOMAIN\\_NUM](#) 0x08
- #define [VTSS\\_PHY\\_TS\\_FIFO\\_SIG\\_SOURCE\\_PORT\\_ID](#) 0x10
- #define [VTSS\\_PHY\\_TS\\_FIFO\\_SIG\\_SEQ\\_ID](#) 0x20
- #define [VTSS\\_PHY\\_TS\\_FIFO\\_SIG\\_DEST\\_MAC](#) 0x40
- #define [VTSS\\_PHY\\_TS\\_SIG\\_LEN](#) 16
- #define [VTSS\\_PHY\\_TS\\_SIG\\_TIME\\_STAMP\\_LEN](#) 10
- #define [VTSS\\_PHY\\_TS\\_SIG\\_DOMAIN\\_NUM\\_LEN](#) 1
- #define [VTSS\\_PHY\\_TS\\_SIG\\_SEQ\\_ID\\_LEN](#) 1
- #define [VTSS\\_PHY\\_TS\\_SIG\\_MSG\\_TYPE\\_LEN](#) 1
- #define [VTSS\\_PHY\\_TS\\_SIG\\_SOURCE\\_PORT\\_ID\\_LEN](#) 10
- #define [VTSS\\_PHY\\_TS\\_SIG\\_SEQUENCE\\_ID\\_LEN](#) 2
- #define [VTSS\\_PHY\\_TS\\_SIG\\_DEST\\_IP\\_LEN](#) 4
- #define [VTSS\\_PHY\\_TS\\_SIG\\_SRC\\_IP\\_LEN](#) 4
- #define [VTSS\\_PHY\\_TS\\_SIG\\_DEST\\_MAC\\_LEN](#) 6
- #define [VTSS\\_PTP\\_IP\\_1588\\_VERSION\\_2\\_1](#) 0x21  
*Defines 1588 version code for Gen-2.1.*
- #define [VTSS\\_PHY\\_TS\\_ETH\\_ADDR\\_MATCH\\_48BIT](#) ((u8)0x01)
- #define [VTSS\\_PHY\\_TS\\_ETH\\_ADDR\\_MATCH\\_ANY\\_UNICAST](#) ((u8)0x02)

- #define VTSS\_PHY\_TS\_ETH\_ADDR\_MATCH\_ANY\_MULTICAST ((u8)0x04)
- #define VTSS\_PHY\_TS\_ETH\_MATCH\_DEST\_ADDR ((u8)0x00)
- #define VTSS\_PHY\_TS\_ETH\_MATCH\_SRC\_ADDR ((u8)0x01)
- #define VTSS\_PHY\_TS\_ETH\_MATCH\_SRC\_OR\_DEST ((u8)0x02)
- #define VTSS\_PHY\_TS\_TAG\_TYPE\_C ((u8)0x01)
- #define VTSS\_PHY\_TS\_TAG\_TYPE\_S ((u8)0x02)
- #define VTSS\_PHY\_TS\_TAG\_TYPE\_I ((u8)0x03)
- #define VTSS\_PHY\_TS\_TAG\_TYPE\_B ((u8)0x04)
- #define VTSS\_PHY\_TS\_TAG\_RANGE\_NONE ((u8)0x00)
- #define VTSS\_PHY\_TS\_TAG\_RANGE\_OUTER ((u8)0x01)
- #define VTSS\_PHY\_TS\_TAG\_RANGE\_INNER ((u8)0x02)
- #define VTSS\_PHY\_TS\_IP\_VER\_4 0x01
- #define VTSS\_PHY\_TS\_IP\_VER\_6 0x02
- #define VTSS\_PHY\_TS\_IP\_MATCH\_SRC 0x00
- #define VTSS\_PHY\_TS\_IP\_MATCH\_DEST 0x01
- #define VTSS\_PHY\_TS\_IP\_MATCH\_SRC\_OR\_DEST 0x02
- #define VTSS\_PHY\_TS\_MPLS\_STACK\_DEPTH\_1 ((u8)0x01)
- #define VTSS\_PHY\_TS\_MPLS\_STACK\_DEPTH\_2 ((u8)0x02)
- #define VTSS\_PHY\_TS\_MPLS\_STACK\_DEPTH\_3 ((u8)0x04)
- #define VTSS\_PHY\_TS\_MPLS\_STACK\_DEPTH\_4 ((u8)0x08)
- #define VTSS\_PHY\_TS\_MPLS\_STACK\_REF\_POINT\_TOP 0x00
- #define VTSS\_PHY\_TS\_MPLS\_STACK\_REF\_POINT\_END 0x01
- #define VTSS\_PHY\_TS\_ENG\_FLOW\_VALID\_FOR\_CH0 0x01

*Port to flow mapping within analyzer engine.*

- #define VTSS\_PHY\_TS\_ENG\_FLOW\_VALID\_FOR\_CH1 0x02
- #define VTSS\_PHY\_TS\_INGR\_ENGINE\_ERR 0x01

*Timestamp interrupt events.*

- #define VTSS\_PHY\_TS\_INGR\_RW\_PREAM\_ERR 0x02
- #define VTSS\_PHY\_TS\_INGR\_RW\_FCS\_ERR 0x04
- #define VTSS\_PHY\_TS\_EGR\_ENGINE\_ERR 0x08
- #define VTSS\_PHY\_TS\_EGR\_RW\_FCS\_ERR 0x10
- #define VTSS\_PHY\_TS\_EGR\_TIMESTAMP\_CAPTURED 0x20
- #define VTSS\_PHY\_TS\_EGR\_FIFO\_OVERFLOW 0x40
- #define VTSS\_PHY\_TS\_DATA\_IN\_RSRVD\_FIELD 0x80
- #define VTSS\_PHY\_TS\_LTC\_NEW\_PPS\_INTRPT 0x100
- #define VTSS\_PHY\_TS\_LTC\_LOAD\_SAVE\_NEW\_TOD 0x200
- #define VTSS\_PHY\_TS\_8487\_XAUI\_SEL\_0 0x01

*Define the Channel selection for 8487.*

- #define VTSS\_PHY\_TS\_8487\_XAUI\_SEL\_1 0x02
- #define VTSS\_PHY\_TS\_INGR\_DATAPATH\_RESET 0x01

*1588 block reset*

- #define VTSS\_PHY\_TS\_EGR\_DATAPATH\_RESET 0x02
- #define VTSS\_PHY\_TS\_INGR\_LTC1\_RESET 0x04
- #define VTSS\_PHY\_TS\_EGR\_LTC2\_RESET 0x08
- #define VTSS\_PHY\_TS\_EGR\_FIFO\_RESET 0x10

## Typedefs

- **typedef struct vtss\_phy\_ts\_alt\_clock\_mode\_s vtss\_phy\_ts\_alt\_clock\_mode\_t**  
*parameter for setting the alternative clock mode.*
- **typedef struct vtss\_phy\_ts\_pps\_config\_s vtss\_phy\_ts\_pps\_conf\_t**  
*PPS Configuration.*
- **typedef i64 vtss\_phy\_ts\_scaled\_ppb\_t**  
*Data type defines the clock frequency ratio in scaled ppb.*
- **typedef struct vtss\_phy\_ltc\_freq\_synth\_s vtss\_phy\_ltc\_freq\_synth\_t**  
*Frequency synthesis pulse configuration.*
- **typedef u32 vtss\_phy\_ts\_fifo\_sig\_mask\_t**
- **typedef void(\* vtss\_phy\_ts\_fifo\_read) (const vtss\_inst\_t inst, const vtss\_port\_no\_t port\_no, const vtss\_phy\_timestamp\_t \*const ts, const vtss\_phy\_ts\_fifo\_sig\_t \*const sig, void \*ctxt, const vtss\_phy\_ts\_fifo\_status\_t status)**  
*Tx TSFIFO read callback function prototype.*
- **typedef u8 vtss\_phy\_ts\_engine\_channel\_map\_t**
- **typedef u32 vtss\_phy\_ts\_event\_t**
- **typedef u32 vtss\_phy\_ts\_8487\_xaui\_sel\_t**
- **typedef u32 vtss\_phy\_ts\_soft\_reset\_t**

## Enumerations

- **enum vtss\_phy\_ts\_todadj\_status\_t { VTSS\_PHY\_TS\_TODADJ\_INPROGRESS, VTSS\_PHY\_TS\_TODADJ\_DONE, VTSS\_PHY\_TS\_TODADJ\_FAIL }**  
*parameter describing various Tx TSFIFO status.*
- **enum vtss\_phy\_ts\_fifo\_status\_t { VTSS\_PHY\_TS\_FIFO\_SUCCESS, VTSS\_PHY\_TS\_FIFO\_OVERFLOW }**  
*parameter describing various Tx TSFIFO status.*
- **enum vtss\_phy\_ts\_encap\_t { VTSS\_PHY\_TS\_ENCAP\_ETH\_PTP, VTSS\_PHY\_TS\_ENCAP\_ETH\_IP\_PTP, VTSS\_PHY\_TS\_ENCAP\_↔ETH\_IP\_IP\_PTP, VTSS\_PHY\_TS\_ENCAP\_ETH\_ETH\_PTP, VTSS\_PHY\_TS\_ENCAP\_ETH\_MPLS\_IP\_PTP, VTSS\_PHY\_↔TS\_ENCAP\_ETH\_MPLS\_ETH\_PTP, VTSS\_PHY\_TS\_ENCAP\_ETH\_MPLS\_ETH\_OAM, VTSS\_PHY\_T↔S\_ENCAP\_ETH\_ETH\_OAM, VTSS\_PHY\_TS\_ENCAP\_ETH\_MPLS\_ETH\_OAM, VTSS\_PHY\_TS\_ENCAP\_ETH\_MPLS\_ACH\_OAM, VTSS\_PHY\_TS\_ENCAP\_ANY, VTSS\_PHY\_TS\_EN↔CAP\_ETH\_GEN, VTSS\_PHY\_TS\_ENCAP\_NONE }**  
*Analyzer supported frame encapsulation type.*
- **enum vtss\_phy\_ts\_engine\_t { VTSS\_PHY\_TS\_PTP\_ENGINE\_ID\_0, VTSS\_PHY\_TS\_PTP\_ENGINE\_ID\_1, VTSS\_PHY\_TS\_OAM\_ENGINE\_ID\_2A, VTSS\_PHY\_TS\_OAM\_ENGINE\_ID\_2B, VTSS\_PHY\_TS\_ENGINE\_ID\_INVALID }**  
*Defines Analyzer engine ID.*
- **enum vtss\_phy\_ts\_engine\_flow\_match\_t { VTSS\_PHY\_TS\_ENG\_FLOW\_MATCH\_ANY, VTSS\_PHY\_TS\_ENG\_FLOW\_MATCH\_...**  
*Flow matching within an analyzer engine.*
- **enum vtss\_phy\_ts\_ptp\_clock\_mode\_t { VTSS\_PHY\_TS\_PTP\_CLOCK\_MODE\_BC1STEP, VTSS\_PHY\_TS\_PTP\_CLOCK\_MODE\_BC2STEP, VTSS\_PHY\_TS\_PTP\_CLOCK\_MODE\_TC1STEP, VTSS\_PHY\_TS\_PTP\_CLOCK\_MODE\_TC2STEP, VTSS\_PHY\_TS\_PTP\_DELAY\_COMP\_ENGINE }**  
*PTP Timestamp Engine operational modes.*
- **enum vtss\_phy\_ts\_ptp\_delaym\_type\_t { VTSS\_PHY\_TS\_PTP\_DELAYM\_P2P, VTSS\_PHY\_TS\_PTP\_DELAYM\_E2E }**  
*PTP delay measurement method.*

- enum `vtss_phy_ts_y1731_oam_delaym_type_t` { `VTSS_PHY_TS_Y1731_OAM_DELAYM_1DM`, `VTSS_PHY_TS_Y1731_OAM_DELAYM_2DM` }

*Y.1731 OAM delay measurement method.*

- enum `vtss_phy_ts_ietf_mpls_ach_oam_delaym_type_t` { `VTSS_PHY_TS_IETF_MPLS_ACH_OAM_DELAYM_DMM`, `VTSS_PHY_TS_IETF_MPLS_ACH_OAM_DELAYM_LDM` }

*IETF MPLS ACH, OAM delay measurement method.*

- enum `vtss_phy_ts_ietf_mpls_ach_oam_ts_format_t` { `VTSS_PHY_TS_IETF_MPLS_ACH_OAM_TS_FORMAT_PTP` = `0x3` }

*Analyzer OAM comparator, IETF MPLS ACH OAM Supported Timestamp Format.*

- enum `vtss_phy_ts_action_format` { `VTSS_PHY_TS_4_BYTE`, `VTSS_PHY_TS_10_BYTE` }

*Timestamp format to be configured in action configuration.*

- enum `vtss_phy_ts_clockfreq_t` { `VTSS_PHY_TS_CLOCK_FREQ_125M`, `VTSS_PHY_TS_CLOCK_FREQ_15625M`, `VTSS_PHY_TS_CLOCK_FREQ_200M`, `VTSS_PHY_TS_CLOCK_FREQ_250M`, `VTSS_PHY_TS_CLOCK_FREQ_500M`, `VTSS_PHY_TS_CLOCK_FREQ_MAX` }

*Timestamp block clock frequencies.*

- enum `vtss_phy_ts_clock_src_t` { `VTSS_PHY_TS_CLOCK_SRC_EXTERNAL`, `VTSS_PHY_TS_CLOCK_SRC_CLIENT_RX`, `VTSS_PHY_TS_CLOCK_SRC_CLIENT_TX`, `VTSS_PHY_TS_CLOCK_SRC_LINE_RX`, `VTSS_PHY_TS_CLOCK_SRC_LINE_TX`, `VTSS_PHY_TS_CLOCK_SRC_INTERNAL` }

*Clock input source.*

- enum `vtss_phy_ts_rxtimestamp_pos_t` { `VTSS_PHY_TS_RX_TIMESTAMP_POS_IN_PTP`, `VTSS_PHY_TS_RX_TIMESTAMP_POS_IN_TSIF` }

*Defines Rx Timestamp position inside PTP frame.*

- enum `vtss_phy_ts_rxtimestamp_len_t` { `VTSS_PHY_TS_RX_TIMESTAMP_LEN_30BIT`, `VTSS_PHY_TS_RX_TIMESTAMP_LEN_32BIT` }

*Defines RX Timestamp format i.e. 30bit or 32bit Rx timestamp.*

- enum `vtss_phy_ts_fifo_mode_t` { `VTSS_PHY_TS_FIFO_MODE_NORMAL`, `VTSS_PHY_TS_FIFO_MODE_SPI` }

*Defines Tx TSFIFO access mode.*

- enum `vtss_phy_ts_fifo_timestamp_len_t` { `VTSS_PHY_TS_FIFO_TIMESTAMP_LEN_4BYTE`, `VTSS_PHY_TS_FIFO_TIMESTAMP_LEN_10BYTE` }

*Defines 4 bytes vs. the default 10 bytes Timestamp stored in Tx TSFIFO.*

- enum `vtss_phy_ts_tc_op_mode_t` { `VTSS_PHY_TS_TC_OP_MODE_A` = `1`, `VTSS_PHY_TS_TC_OP_MODE_B` = `0`, `VTSS_PHY_TS_TC_OP_MODE_C` = `2` }

*Defines the Transparent Clock Operating Mode.*

- enum `vtss_phy_ts_nphase_sampler_t` { `VTSS_PHY_TS_NPHASE_PPS_O`, `VTSS_PHY_TS_NPHASE_PPS_RI`, `VTSS_PHY_TS_NPHASE_EGR_SOF`, `VTSS_PHY_TS_NPHASE_ING_SOF`, `VTSS_PHY_TS_NPHASE_LS`, `VTSS_PHY_TS_NPHASE_MAX` }

*enum for n-phase samplers*

- enum `vtss_phy_ts_ptp_message_type_t` { `PTP_SYNC_MSG`, `PTP_DELAY_REQ_MSG`, `PTP_PDELAY_REQ_MSG`, `PTP_PDELAY_RESP_MSG` }

*PTP Event Message TYPES.*

## Functions

- `vtss_rc vtss_phy_ts_alt_clock_saved_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `u32` \*const saved)

*Get the latest saved nanosec counter from the alternative clock.*

- `vtss_rc vtss_phy_ts_alt_clock_mode_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_ts_alt_clock_mode_t` \*const phy\_alt\_clock\_mode)

- `vtss_rc vtss_phy_ts_alt_clock_mode_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_ts_alt_clock_mode_t` \*const phy\_alt\_clock\_mode)
 

*Set the alternative clock mode. This function configures the loopbacks.*
- `vtss_rc vtss_phy_ts_pps_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_ts_pps_conf_t` \*const phy\_pps\_conf)
 

*Set offset for the PPS generation.*
- `vtss_rc vtss_phy_ts_pps_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_ts_pps_conf_t` \*const phy\_pps\_conf)
 

*Get offset for the PPS generation.*
- `vtss_rc vtss_phy_ts_ingress_latency_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_timeinterval_t` \*const latency)
 

*Set the ingress latency.*
- `vtss_rc vtss_phy_ts_ingress_latency_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_timeinterval_t` \*const latency)
 

*Get the ingress latency.*
- `vtss_rc vtss_phy_ts_egress_latency_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_timeinterval_t` \*const latency)
 

*Set the egress latency.*
- `vtss_rc vtss_phy_ts_egress_latency_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_timeinterval_t` \*const latency)
 

*Get the egress latency.*
- `vtss_rc vtss_phy_ts_path_delay_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_timeinterval_t` \*const path\_delay)
 

*Set the path delay.*
- `vtss_rc vtss_phy_ts_path_delay_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_timeinterval_t` \*const path\_delay)
 

*Get the path delay.*
- `vtss_rc vtss_phy_ts_delay_asymmetry_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_timeinterval_t` \*const delay\_asym)
 

*Set the delay asymmetry.*
- `vtss_rc vtss_phy_ts_delay_asymmetry_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_timeinterval_t` \*const delay\_asym)
 

*Get the delay asymmetry.*
- `vtss_rc vtss_phy_ts_ptptime_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_timestamp_t` \*const ts)
 

*Set the current PTP time into the PHY.*
- `vtss_rc vtss_phy_ts_ptptime_set_done` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no)
 

*Setting of the current PTP time into the PHY is completed.*
- `vtss_rc vtss_phy_ts_ptptime_arm` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no)
 

*Arm the local time of the PHY so that in next pps it can be read.*
- `vtss_rc vtss_phy_ts_ptptime_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_timestamp_t` \*const ts)
 

*Get the armed PTP time from the PHY.*
- `vtss_rc vtss_phy_ts_load_ptptime_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_timestamp_t` \*const ts)
 

*Get the PTP time from the PHY load registers.*
- `vtss_rc vtss_phy_ts_sertod_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_ts_sertod_conf_t` \*const sertod\_conf)
 

*Set Enable/Disable Serial ToD.*
- `vtss_rc vtss_phy_ts_sertod_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_ts_sertod_conf_t` \*const sertod\_conf)
 

*Get Enable/Disable Serial ToD.*

- `vtss_rc vtss_phy_ts_loadpulse_delay_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `u16` \*const delay)
 

*Set load pulse delay.*
- `vtss_rc vtss_phy_ts_loadpulse_delay_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `u16` \*const delay)
 

*Get load pulse delay.*
- `vtss_rc vtss_phy_ts_clock_rateadj_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_ts_scaled_ppb_t` \*const adj)
 

*Adjust the local clock rate.*
- `vtss_rc vtss_phy_ts_clock_rateadj_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_ts_scaled_ppb_t` \*const adj)
 

*Get the clock rate adjustment value.*
- `vtss_rc vtss_phy_ts_clock_rateadj_ppm_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_ts_scaled_ppb_t` \*const adj)
 

*Adjust ppm of the local clock rate .*
- `vtss_rc vtss_phy_ts_clock_rateadj_ppm_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_ts_scaled_ppb_t` \*const adj)
 

*Get the clock rate ppm adjustment value.*
- `vtss_rc vtss_phy_ts_ptptime_adj1ns` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `BOOL` incr)
 

*Increment/decrement the LTC clock value by 1 ns.*
- `vtss_rc vtss_phy_ts_timeofday_offset_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `i32` offset)
 

*Subtract offset from the current time.*
- `vtss_rc vtss_phy_ts_ongoing_adjustment` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_ts_todadj_status_t` \*const ongoing\_adjustment)
 

*Return the status of the LTC time adjustment.*
- `vtss_rc vtss_phy_ts_ltc_freq_synth_pulse_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_ltc_freq_synth_t` \*const ltc\_freq\_synthesis)
 

*Return the status of the LTC time adjustment.*
- `vtss_rc vtss_phy_ts_ltc_freq_synth_pulse_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_ltc_freq_synth_t` \*const ltc\_freq\_synthesis)
 

*Return the status of the LTC time adjustment.*
- `vtss_rc vtss_phy_daisy_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_daisy_chain_conf_t` \*const daisy\_chain)
 

*configure the daisy chain for TS FIFO*
- `vtss_rc vtss_phy_daisy_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_daisy_chain_conf_t` \*const daisy\_chain)
 

*getting the daisy chain for TS FIFO*
- `vtss_rc vtss_phy_ts_fifo_sig_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_ts_fifo_sig_mask_t` sig\_mask)
 

*Set Tx TSFIFO i.e. frame signature in Tx TSFIFO.*
- `vtss_rc vtss_phy_ts_fifo_sig_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_ts_fifo_sig_mask_t` \*const sig\_mask)
 

*Get frame signature mask in Tx TSFIFO.*
- `vtss_rc vtss_phy_ts_fifo_empty` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no)
 

*Read timestamp from Tx TSFIFO.*
- `vtss_rc vtss_phy_ts_fifo_read_install` (const `vtss_inst_t` inst, `vtss_phy_ts_fifo_read` rd\_cb, void \*cctxt)
 

*Install callback to read data (signature + timestamp) from Tx TSFIFO.*
- `vtss_rc vtss_phy_ts_fifo_read_cb_get` (const `vtss_inst_t` inst, `vtss_phy_ts_fifo_read` \*rd\_cb, void \*\*cctxt)
 

*Get the fifo read callback function installed.*
- `vtss_rc vtss_phy_ts_ingress_engine_init` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_ts_engine_t` eng\_id, const `vtss_phy_ts_encap_t` encaps\_type, const `u8` flow\_st\_index, const `u8` flow\_end\_index, const `vtss_phy_ts_engine_flow_match_t` flow\_match\_mode)

*Initialize an analyzer ingress engine for an encapsulation type.*

- `vtss_rc vtss_phy_ts_ingress_engine_init_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_ts_engine_t` eng\_id, `vtss_phy_ts_eng_init_conf_t` \*const init\_conf)

*Get the configuration parameters passed in engine\_init of an analyzer ingress engine for a specific engine ID.*

- `vtss_rc vtss_phy_ts_ingress_engine_clear` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_ts_engine_t` eng\_id)

*Clear/release an analyzer ingress engine already initialized.*

- `vtss_rc vtss_phy_ts_egress_engine_init` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_ts_engine_t` eng\_id, const `vtss_phy_ts_encap_t` encapsulation\_type, const `u8` flow\_st\_index, const `u8` flow\_end\_index, const `vtss_phy_ts_engine_flow_match_t` flow\_match\_mode)

*Initialize an analyzer egress engine for an encapsulation type.*

- `vtss_rc vtss_phy_ts_egress_engine_init_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_ts_engine_t` eng\_id, `vtss_phy_ts_eng_init_conf_t` \*const init\_conf)

*Get the configuration parameters passed in engine\_init of an analyzer egress engine for a specific engine ID.*

- `vtss_rc vtss_phy_ts_egress_engine_clear` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_ts_engine_t` eng\_id)

*Clear/release an analyzer egress engine already initialized.*

- `vtss_rc vtss_phy_ts_ingress_engine_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_ts_engine_t` eng\_id, const `vtss_phy_ts_engine_flow_conf_t` \*const flow\_conf)

*Configure ingress analyzer flow.*

- `vtss_rc vtss_phy_ts_ingress_engine_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_ts_engine_t` eng\_id, `vtss_phy_ts_engine_flow_conf_t` \*const flow\_conf)

*Get ingress analyzer flow.*

- `vtss_rc vtss_phy_ts_egress_engine_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_ts_engine_t` eng\_id, const `vtss_phy_ts_engine_flow_conf_t` \*const flow\_conf)

*Configure egress analyzer flow.*

- `vtss_rc vtss_phy_ts_egress_engine_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_ts_engine_t` eng\_id, `vtss_phy_ts_engine_flow_conf_t` \*const flow\_conf)

*Get egress analyzer flow.*

- `vtss_rc vtss_phy_ts_ingress_engine_action_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_ts_engine_t` eng\_id, const `vtss_phy_ts_engine_action_t` \*const action\_conf)

*Configure ingress analyzer engine action.*

- `vtss_rc vtss_phy_ts_ingress_engine_action_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_ts_engine_t` eng\_id, `vtss_phy_ts_engine_action_t` \*const action\_conf)

*Get ingress analyzer engine action. For multi-port timestamp block, either of the port can be used to get the ingress engine.*

- `vtss_rc vtss_phy_ts_egress_engine_action_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_ts_engine_t` eng\_id, const `vtss_phy_ts_engine_action_t` \*const action\_conf)

*Configure egress analyzer engine action.*

- `vtss_rc vtss_phy_ts_egress_engine_action_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_ts_engine_t` eng\_id, `vtss_phy_ts_engine_action_t` \*const action\_conf)

*Get egress analyzer engine action. For multi-port timestamp block, either of the port can be used to get the egress engine.*

- `vtss_rc vtss_phy_ts_event_enable_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `BOOL` enable, const `vtss_phy_ts_event_t` ev\_mask)

*Enabling / Disabling of events.*

- `vtss_rc vtss_phy_ts_event_enable_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_ts_event_t` \*const ev\_mask)

*Get Enabling of events.*

- `vtss_rc vtss_phy_ts_event_poll` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_ts_event_t` \*const status)

*Polling function called at by interrupt or periodically.*

- `vtss_rc vtss_phy_ts_stats_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_ts_stats_t` \*const statistics)

- Get Timestamp statistics.*
- `vtss_rc vtss_phy_ts_correction_overflow_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `BOOL` \*const ingr\_overflow, `BOOL` \*const egr\_overflow)

*Get the correction field overflow status in ingress and egress.*

  - `vtss_rc vtss_phy_ts_mode_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `BOOL` enable)

*Enable/disable timestamp block.*

  - `vtss_rc vtss_phy_ts_mode_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `BOOL` \*const enable)

*Get timestamp block status i.e. enable/disable.*

  - `vtss_rc vtss_phy_ts_init` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_ts_init_conf_t` \*const conf)

*Init timestamp block.*

  - `vtss_rc vtss_phy_ts_init_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `BOOL` \*const init\_done, `vtss_phy_ts_init_conf_t` \*const conf)

*Get the timestamp init config parameters.*

  - `vtss_rc vtss_phy_ts_nphase_status_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_ts_nphase_sampler_t` sampler, `vtss_phy_ts_nphase_status_t` \*const status)

*Get N-Phase sampler status.*

  - `vtss_rc vtss_phy_ts_hiacc_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_ts_nphase_sampler_t` sampler, const `BOOL` enable)

*Enable N-Phase sampler.*

  - `vtss_rc vtss_phy_ts_hiacc_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_ts_nphase_sampler_t` sampler, `BOOL` const \*enable)

*N-Phase sampler status get.*

  - `vtss_rc vtss_phy_ts_block_soft_reset` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_ts_soft_reset_t` ts\_reset)

*reset 1588 block.*

  - `vtss_rc vtss_phy_ts_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.*

### 6.32.1 Detailed Description

PHY TimeStamping API.

This header file describes PTP/OAM TimeStamping API functions

### 6.32.2 Macro Definition Documentation

#### 6.32.2.1 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.

#### 6.32.2.2 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.

#### 6.32.2.3 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.

#### 6.32.2.4 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.

#### 6.32.2.5 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.

#### 6.32.2.6 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.

#### 6.32.2.7 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.

#### 6.32.2.8 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.

### 6.32.2.9 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.

### 6.32.2.10 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.

### 6.32.2.11 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.

### 6.32.2.12 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.

### 6.32.2.13 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.

### 6.32.2.14 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.

### 6.32.2.15 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.

### 6.32.2.16 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.

### 6.32.2.17 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.

### 6.32.2.18 VTSS\_PTP\_IP\_1588\_VERSION\_2\_1

```
#define VTSS_PTP_IP_1588_VERSION_2_1 0x21
```

Defines 1588 version code for Gen2.1.

1588 block version for Gen2.1

Definition at line 605 of file vtss\_phy\_ts\_api.h.

**6.32.2.19 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.

**6.32.2.20 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.

**6.32.2.21 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.

**6.32.2.22 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.

**6.32.2.23 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.

### 6.32.2.24 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.

### 6.32.2.25 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.

### 6.32.2.26 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.

### 6.32.2.27 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.

### 6.32.2.28 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.

### 6.32.2.29 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.

### 6.32.2.30 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.

### 6.32.2.31 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.

### 6.32.2.32 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.

### 6.32.2.33 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.

### 6.32.2.34 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.

### 6.32.2.35 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.

### 6.32.2.36 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.

### 6.32.2.37 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.

### 6.32.2.38 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.

### 6.32.2.39 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.

### 6.32.2.40 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.

### 6.32.2.41 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.

### 6.32.2.42 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.

### 6.32.2.43 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.

**6.32.2.44 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.

**6.32.2.45 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.

**6.32.2.46 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.

**6.32.2.47 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.

**6.32.2.48 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.

**6.32.2.49 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.

**6.32.2.50 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.

**6.32.2.51 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.

**6.32.2.52 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.

**6.32.2.53 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.

**6.32.2.54 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.

**6.32.2.55 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.

**6.32.2.56 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.

**6.32.2.57 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.

**6.32.2.58 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.

### 6.32.2.59 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.

### 6.32.2.60 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.

### 6.32.2.61 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.

## 6.32.3 Typedef Documentation

### 6.32.3.1 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.

### 6.32.3.2 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.

### 6.32.3.3 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.

### 6.32.3.4 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.

### 6.32.3.5 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.

### 6.32.3.6 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.

### 6.32.3.7 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.

### 6.32.3.8 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.

## 6.32.4 Enumeration Type Documentation

### 6.32.4.1 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.

### 6.32.4.2 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.

### 6.32.4.3 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.

### 6.32.4.4 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.

### 6.32.4.5 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.

#### 6.32.4.6 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.

#### 6.32.4.7 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.

**6.32.4.8 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.

**6.32.4.9 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.

#### 6.32.4.10 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.

#### 6.32.4.11 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.

#### 6.32.4.12 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.

#### 6.32.4.13 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.

#### 6.32.4.14 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.

## 6.32.4.15 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.

## 6.32.4.16 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.

## 6.32.4.17 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.

### 6.32.4.18 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`.

### 6.32.4.19 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`.

### 6.32.4.20 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.

## 6.32.5 Function Documentation

### 6.32.5.1 vtss\_phy\_ts\_alt\_clock\_saved\_get()

```
vtss_rc vtss_phy_ts_alt_clock_saved_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    u32 *const saved )
```

Get the latest saved nanosec counter from the alternative clock.

#### Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>saved</i>	[OUT] latest saved value.

#### Returns

Return code.

### 6.32.5.2 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.

**6.32.5.3 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.

**6.32.5.4 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.

**6.32.5.5 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.

**6.32.5.6 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.

**6.32.5.7 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.

**6.32.5.8 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.

**6.32.5.9 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.

**6.32.5.10 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.

**6.32.5.11 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.

**6.32.5.12 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.

**6.32.5.13 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.

**6.32.5.14 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.

### 6.32.5.15 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.

### 6.32.5.16 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.

### 6.32.5.17 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.

### 6.32.5.18 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.

### 6.32.5.19 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.

**6.32.5.20 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.

**6.32.5.21 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.

**6.32.5.22 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.

**6.32.5.23 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.

### 6.32.5.24 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.

### 6.32.5.25 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.

### 6.32.5.26 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.

**6.32.5.27 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.

**6.32.5.28 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.

**6.32.5.29 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.

**6.32.5.30 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.

### 6.32.5.31 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.

### 6.32.5.32 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.

### 6.32.5.33 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.

**6.32.5.34 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.

**6.32.5.35 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.

**6.32.5.36 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.

**6.32.5.37 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.

**6.32.5.38 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.

**6.32.5.39 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.

**6.32.5.40 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.

**6.32.5.41 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.

### 6.32.5.42 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.

### 6.32.5.43 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.

### 6.32.5.44 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.

### 6.32.5.45 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.

### 6.32.5.46 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.

**6.32.5.47 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.

**6.32.5.48 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.

**6.32.5.49 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.

**6.32.5.50 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.

**6.32.5.51 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.

**6.32.5.52 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.

**6.32.5.53 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.

**6.32.5.54 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.

**6.32.5.55 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.

**6.32.5.56 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.

### 6.32.5.57 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.

### 6.32.5.58 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.

## 6.32.5.59 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.

## 6.32.5.60 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.

## 6.32.5.61 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.

### 6.32.5.62 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.

### 6.32.5.63 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.

**6.32.5.64 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.

**6.32.5.65 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.

**6.32.5.66 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.

**6.32.5.67 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.

**6.32.5.68 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.

**6.32.5.69 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.

**6.32.5.70 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.

**6.32.5.71 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.

### 6.32.5.72 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.

### 6.32.5.73 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.

### 6.32.5.74 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.

**6.32.5.75 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.

## 6.32.5.76 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.

## 6.32.5.77 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.

## 6.33 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 10240

### Enumerations

- enum [vtss\\_miim\\_controller\\_t](#) {  
 VTSS\_MIIM\_CONTROLLER\_0 = 0, VTSS\_MIIM\_CONTROLLER\_1 = 1, VTSS\_MIIM\_CONTROLLER\_2 =  
 2, VTSS\_MIIM\_CONTROLLERS,  
 VTSS\_MIIM\_CONTROLLER\_NONE = -1 }  
*MII management controller.*
- enum [vtss\\_internal\\_bw\\_t](#) { VTSS\_BW\_DEFAULT, VTSS\_BW\_1G, VTSS\_BW\_2500M, VTSS\_BW\_UNDEFINED  
 }  
*The internal bandwidth allocated for the port.*

- enum `vtss_port_clause_37_remote_fault_t` { `VTSS_PORT_CLAUSE_37_RF_LINK_OK` = ((0<<1) | (0<<0)), `VTSS_PORT_CLAUSE_37_RF_OFFLINE` = ((1<<1) | (0<<0)), `VTSS_PORT_CLAUSE_37_RF_LINK_FAILURE` = ((0<<1) | (1<<0)), `VTSS_PORT_CLAUSE_37_RF_AUTONEG_ERROR` = ((1<<1) | (1<<0)) }

*Auto-negotiation remote fault type.*

- enum `vtss_port_max_tags_t` { `VTSS_PORT_MAX_TAGS_NONE`, `VTSS_PORT_MAX_TAGS_ONE`, `VTSS_PORT_MAX_TAGS_TWO` }

*VLAN awareness for frame length check.*

- enum `vtss_port_loop_t` { `VTSS_PORT_LOOP_DISABLE`, `VTSS_PORT_LOOP_PCS_HOST` }

*Port loop back configuration.*

- enum `vtss_port_forward_t` { `VTSS_PORT_FORWARD_ENABLED`, `VTSS_PORT_FORWARD_DISABLED`, `VTSS_PORT_FORWARD_INGRESS`, `VTSS_PORT_FORWARD_EGRESS` }

*Port forwarding state.*

## Functions

- `vtss_rc vtss_port_map_set` (const `vtss_inst_t` inst, const `vtss_port_map_t` port\_map[`VTSS_PORT_ARRAY_SIZE`])  
*Set port map.*
- `vtss_rc vtss_port_map_get` (const `vtss_inst_t` inst, `vtss_port_map_t` port\_map[`VTSS_PORT_ARRAY_SIZE`])  
*Get port map.*
- `vtss_rc vtss_port_clause_37_control_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_port_clause_37_control_t` \*const control)  
*Get clause 37 auto-negotiation Control word.*
- `vtss_rc vtss_port_clause_37_control_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_port_clause_37_control_t` \*const control)  
*Set clause 37 auto-negotiation Control word.*
- `vtss_rc vtss_port_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_port_conf_t` \*const conf)  
*Set port configuration. Note: If if\_type in the vtss\_port\_conf\_t/vtss\_port\_interface\_t definition is set to VTSS\_PORT\_INTERFACE\_QSGMII, the ports are mapped together in groups of four. If one of the four ports is used, all four ports in the group must always be configured, but the four ports doesn't need to be configured with the same configuration. This is needed in order to achieve correct comma alignment at the QSGMII interface. Which ports that are mapped together can be found in the chip data-sheet.*
- `vtss_rc vtss_port_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_port_conf_t` \*const conf)  
*Get port setup.*
- `vtss_rc vtss_port_status_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_port_status_t` \*const status)  
*Get port status.*
- `vtss_rc vtss_port_counters_update` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no)  
*Update counters for port.*
- `vtss_rc vtss_port_counters_clear` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no)  
*Clear counters for port.*
- `vtss_rc vtss_port_counters_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_port_counters_t` \*const counters)  
*Get counters for port.*
- `vtss_rc vtss_port_basic_counters_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_basic_counters_t` \*const counters)  
*Get basic counters for port.*
- `vtss_rc vtss_port_forward_state_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_port_forward_t` \*const forward)  
*Get port forwarding state.*

- `vtss_rc vtss_port_forward_state_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_port_forward_t` forward)
 

*Set port forwarding state.*
- `vtss_rc vtss_port_ifh_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_port_ifh_t` \*const conf)
 

*Set port Internal Frame Header settings.*
- `vtss_rc vtss_port_ifh_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_port_ifh_t` \*const conf)
 

*Get port Internal Frame Header settings.*
- `vtss_rc vtss_miim_read` (const `vtss_inst_t` inst, const `vtss_chip_no_t` chip\_no, const `vtss_miim_controller_t` miim\_controller, const `u8` miim\_addr, const `u8` addr, `u16` \*const value)
 

*Direct MIIM read (bypassing port map)*
- `vtss_rc vtss_miim_write` (const `vtss_inst_t` inst, const `vtss_chip_no_t` chip\_no, const `vtss_miim_controller_t` miim\_controller, const `u8` miim\_addr, const `u8` addr, const `u16` value)
 

*Direct MIIM write (bypassing port map)*
- `vtss_rc vtss_port_mmd_read` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `u8` mmd, const `u16` addr, `u16` \*const value)
 

*Read value from MMD register.*
- `vtss_rc vtss_port_mmd_read_inc` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `u8` mmd, const `u16` addr, `u16` \*const buf, `u8` count)
 

*Read values (a number of 16 bit values) from MMD register.*
- `vtss_rc vtss_port_mmd_write` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `u8` mmd, const `u16` addr, const `u16` value)
 

*Write value to MMD register.*
- `vtss_rc vtss_port_mmd_masked_write` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `u8` mmd, const `u16` addr, const `u16` value, const `u16` mask)
 

*Read, modify and write value to MMD register.*
- `vtss_rc vtss_mmd_read` (const `vtss_inst_t` inst, const `vtss_chip_no_t` chip\_no, const `vtss_miim_controller_t` miim\_controller, const `u8` miim\_addr, const `u8` mmd, const `u16` addr, `u16` \*const value)
 

*Direct MMD read (Clause 45, bypassing port map)*
- `vtss_rc vtss_mmd_write` (const `vtss_inst_t` inst, const `vtss_chip_no_t` chip\_no, const `vtss_miim_controller_t` miim\_controller, const `u8` miim\_addr, const `u8` mmd, const `u16` addr, const `u16` value)
 

*Direct MMD write (Clause 45, bypassing port map)*

### 6.33.1 Detailed Description

Port API.

### 6.33.2 Macro Definition Documentation

#### 6.33.2.1 CHIP\_PORT\_UNUSED

```
#define CHIP_PORT_UNUSED -1
```

Signifies an unused chip port

Definition at line 69 of file vtss\_port\_api.h.

### 6.33.2.2 VTSS\_FRAME\_GAP\_DEFAULT

```
#define VTSS_FRAME_GAP_DEFAULT 0
```

Default frame gap used

Definition at line 203 of file vtss\_port\_api.h.

### 6.33.2.3 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.

### 6.33.2.4 VTSS\_MAX\_FRAME\_LENGTH\_MAX [1/2]

```
#define VTSS_MAX_FRAME_LENGTH_MAX 10240
```

Maximum frame length supported

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

### 6.33.2.5 VTSS\_MAX\_FRAME\_LENGTH\_MAX [2/2]

```
#define VTSS_MAX_FRAME_LENGTH_MAX 10240
```

Maximum frame length supported

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

## 6.33.3 Enumeration Type Documentation

### 6.33.3.1 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_CONTROLLER_2	MIIM controller 2
VTSS_MIIM_CONTROLLERS	Number of MIIM controllers
VTSS_MIIM_CONTROLLER_NONE	Unassigned MIIM controller

Definition at line 42 of file vtss\_port\_api.h.

### 6.33.3.2 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.

### 6.33.3.3 ytss 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.

#### 6.33.3.4 vtss\_port\_max\_tags\_t

enum `vtss_port_max_tags_t`

VLAN awareness for frame length check.

Enumerator

VTSS_PORT_MAX_TAGS_NONE	No extra tags allowed
VTSS_PORT_MAX_TAGS_ONE	Single tag allowed
VTSS_PORT_MAX_TAGS_TWO	Single and double tag allowed

Definition at line 246 of file vtss\_port\_api.h.

#### 6.33.3.5 vtss\_port\_loop\_t

enum `vtss_port_loop_t`

Port loop back configuration.

Enumerator

VTSS_PORT_LOOP_DISABLE	No port loop
VTSS_PORT_LOOP_PCS_HOST	PCS host port loop

Definition at line 254 of file vtss\_port\_api.h.

#### 6.33.3.6 vtss\_port\_forward\_t

enum `vtss_port_forward_t`

Port forwarding state.

Enumerator

VTSS_PORT_FORWARD_ENABLED	Forward in both directions
VTSS_PORT_FORWARD_DISABLED	Forwarding and learning disabled
VTSS_PORT_FORWARD_INGRESS	Forward frames from port only
VTSS_PORT_FORWARD_EGRESS	Forward frames to port only (learning disabled)

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

## 6.33.4 Function Documentation

### 6.33.4.1 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.

### 6.33.4.2 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.

### 6.33.4.3 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.

**6.33.4.4 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.

**6.33.4.5 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.

**6.33.4.6 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.

**6.33.4.7 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.

#### 6.33.4.8 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.

#### 6.33.4.9 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.

#### 6.33.4.10 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.

**6.33.4.11 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.

**6.33.4.12 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.

**6.33.4.13 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.

**6.33.4.14 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.

### 6.33.4.15 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.

### 6.33.4.16 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.

## 6.33.4.17 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.

## 6.33.4.18 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.

#### 6.33.4.19 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.

#### 6.33.4.20 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.

## 6.33.4.21 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.

## 6.33.4.22 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.

**6.33.4.23 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.

**6.34 vtss\_api/include/vtss\_qos\_api.h File Reference**

QoS API.

```
#include <vtss/api/types.h>
```

**Data Structures**

- struct **vtss\_red\_v2\_t**  
*Random Early Detection configuration struct version 2 (per queue, per dpl - switch global)*
- struct **vtss\_qos\_conf\_t**  
*All parameters below are defined per chip.*
- struct **vtss\_policer\_t**  
*Policer.*
- struct **vtss\_policer\_ext\_t**

- Policer Extensions.*
- struct [vtss\\_dlb\\_policer\\_conf\\_t](#)  
*Dual leaky buckets policer configuration.*
  - struct [vtss\\_shaper\\_t](#)  
*Shaper.*
  - struct [vtss\\_qos\\_port\\_conf\\_t](#)  
*QoS setup per port.*
  - struct [vtss\\_qce\\_mac\\_t](#)  
*QCE MAC information.*
  - struct [vtss\\_qce\\_tag\\_t](#)  
*QCE tag information.*
  - struct [vtss\\_qce\\_frame\\_etype\\_t](#)  
*Frame data for VTSS\_QCE\_TYPE\_ETYPE.*
  - struct [vtss\\_qce\\_frame\\_llc\\_t](#)  
*Frame data for VTSS\_QCE\_TYPE\_LLC.*
  - 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\_WRED\_DPL\_CNT 3
- #define VTSS\_WRED\_GROUP\_CNT 3
- #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 [vtss\\_red\\_v2\\_t](#) [vtss\\_red\\_v3\\_t](#)
- typedef [u32](#) [vtss\\_qcl\\_id\\_t](#)  
*QCL ID type.*

## Enumerations

- enum `vtss_wred_v2_max_t` { `VTSS_WRED_V2_MAX_DP`, `VTSS_WRED_V2_MAX_FL` }
 

*Random Early Detection version 2. Select if max means max drop probability or max fill level.*
- 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_TYPE_ETYPE`, `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.*

### 6.34.1 Detailed Description

QoS API.

This header file describes Quality of Service functions

### 6.34.2 Macro Definition Documentation

### 6.34.2.1 VTSS\_WRED\_DPL\_CNT

```
#define VTSS_WRED_DPL_CNT 3
```

Number of dpl's. There are no profile for dpl 0, only for dpl 1 to 3

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

### 6.34.2.2 VTSS\_WRED\_GROUP\_CNT

```
#define VTSS_WRED_GROUP_CNT 3
```

Number of groups

Definition at line 84 of file vtss\_qos\_api.h.

### 6.34.2.3 VTSS\_PORT\_POLICERS

```
#define VTSS_PORT_POLICERS 4
```

Number of port policers

Definition at line 177 of file vtss\_qos\_api.h.

### 6.34.2.4 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.

### 6.34.2.5 VTSS\_QCL\_IDS

```
#define VTSS_QCL_IDS 1
```

Number of QCLs

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

#### 6.34.2.6 VTSS\_QCL\_ID\_START

```
#define VTSS_QCL_ID_START 0
```

QCL ID start number

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

#### 6.34.2.7 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.

#### 6.34.2.8 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.

#### 6.34.2.9 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.

### 6.34.3 Typedef Documentation

#### 6.34.3.1 vtss\_red\_v3\_t

```
typedef vtss_red_v2_t vtss_red_v3_t
```

Version 3 uses the v2 struct (per queue, per dpl, per group - switch global

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

### 6.34.4 Enumeration Type Documentation

#### 6.34.4.1 vtss\_wred\_v2\_max\_t

```
enum vtss_wred_v2_max_t
```

Random Early Detection version 2. Select if max means max drop probability or max fill level.

## Enumerator

VTSS_WRED_V2_MAX_DP	Unit for max is drop probability
VTSS_WRED_V2_MAX_FL	Unit for max is fill level

Definition at line 65 of file vtss\_qos\_api.h.

## 6.34.4.2 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.

## 6.34.4.3 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.

## 6.34.4.4 vtss\_dscpemode\_t

```
enum vtss_dscpemode_t
```

DSCP mode for egress port.

## Enumerator

VTSS_DSCP_EMODE_DISABLE	DSCP not remarked
VTSS_DSCP_EMODE_REMARK	DSCP remarked with DSCP value from analyzer
VTSS_DSCP_EMODE_REMAP	DSCP remarked with DSCP value from analyzer remapped through global remap table

Definition at line 333 of file vtss\_qos\_api.h.

## 6.34.4.5 vtss\_qce\_type\_t

enum [vtss\\_qce\\_type\\_t](#)

QoS Control Entry type.

## Enumerator

VTSS_QCE_TYPE_ANY	Any frame type
VTSS_QCE_TYPE_ETYPE	Ethernet Type
VTSS_QCE_TYPE LLC	LLC
VTSS_QCE_TYPE_SNAP	SNAP
VTSS_QCE_TYPE_IPV4	IPv4
VTSS_QCE_TYPE_IPV6	IPv6

Definition at line 460 of file vtss\_qos\_api.h.

## 6.34.5 Function Documentation

## 6.34.5.1 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.

**6.34.5.2 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.

**6.34.5.3 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.

**6.34.5.4 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.

#### 6.34.5.5 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.

#### 6.34.5.6 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.

**6.34.5.7 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.

**6.35 vtss\_api/include/vtss\_rab\_api.h File Reference**

RAB API.

```
#include <vtss/api/types.h>
```

**6.35.1 Detailed Description**

RAB API.

## 6.36 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\\_ptp\\_action\\_conf\\_t](#)  
*ACL PTP action configuration.*
- struct [vtss\\_acl\\_action\\_t](#)  
*ACL Action.*
- struct [vtss\\_acl\\_port\\_conf\\_t](#)  
*ACL port configuration.*
- struct [vtss\\_ace\\_ptp\\_t](#)  
*PTP header filtering.*
- struct [vtss\\_ace\\_sip\\_smac\\_t](#)  
*SIP/SMAC filtering.*
- 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

## 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\_acl\_port\_action\_t { VTSS\_ACL\_PORT\_ACTION\_NONE, VTSS\_ACL\_PORT\_ACTION\_FILTER, VTSS\_ACL\_PORT\_ACTION\_REDIR }**  
*ACL port action.*
- **enum vtss\_acl\_ptp\_action\_t { VTSS\_ACL\_PTP\_ACTION\_NONE, VTSS\_ACL\_PTP\_ACTION\_ONE\_STEP, VTSS\_ACL\_PTP\_ACTION\_ONE\_STEP\_ADD\_DELAY\_1, VTSS\_ACL\_PTP\_ACTION\_ONE\_STEP\_SUB\_DELAY\_1, VTSS\_ACL\_PTP\_ACTION\_ONE\_STEP\_SUB\_DELAY\_2, VTSS\_ACL\_PTP\_ACTION\_TWO\_STEP }**  
*ACL PTP action.*
- **enum vtss\_acl\_ptp\_rsp\_t { VTSS\_ACL\_PTP\_RSP\_NONE, VTSS\_ACL\_PTP\_RSP\_DLY\_REQ\_RSP\_TS\_UPD, VTSS\_ACL\_PTP\_RSP\_DLY\_REQ\_RSP\_NO\_TS }**  
*ACL PTP response action.*
- **enum vtss\_ace\_type\_t { VTSS\_ACE\_TYPE\_ANY, VTSS\_ACE\_TYPEETYPE, VTSS\_ACE\_TYPE\_LLIC, VTSS\_ACE\_TYPE\_SNAP, VTSS\_ACE\_TYPE\_ARP, VTSS\_ACE\_TYPE\_IPV4, VTSS\_ACE\_TYPE\_IPV6 }**  
*ACE frame type.*
- **enum vtss\_ace\_bit\_t { VTSS\_ACE\_BIT\_ANY, VTSS\_ACE\_BIT\_0, VTSS\_ACE\_BIT\_1 }**  
*ACE 1 bit.*

## Functions

- `vtss_rc vtss_auth_port_state_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_auth_state_t` \*const state)
 

*Get 802.1X Authentication state for a port.*
- `vtss_rc vtss_auth_port_state_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_auth_state_t` state)
 

*Set 802.1X Authentication state for a port.*
- `vtss_rc vtss_acl_policer_conf_get` (const `vtss_inst_t` inst, const `vtss_acl_policer_no_t` policer\_no, `vtss_acl_policer_conf_t` \*const conf)
 

*Get ACL policer configuration.*
- `vtss_rc vtss_acl_policer_conf_set` (const `vtss_inst_t` inst, const `vtss_acl_policer_no_t` policer\_no, const `vtss_acl_policer_conf_t` \*const conf)
 

*Set ACL policer configuration.*
- `vtss_rc vtss_acl_port_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_acl_port_conf_t` \*const conf)
 

*Get ACL configuration for port.*
- `vtss_rc vtss_acl_port_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_acl_port_conf_t` \*const conf)
 

*Set ACL configuration for port.*
- `vtss_rc vtss_acl_port_counter_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_acl_port_counter_t` \*const counter)
 

*Get default action counter for port.*
- `vtss_rc vtss_acl_port_counter_clear` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no)
 

*Clear default action counter for port.*
- `vtss_rc vtss_ace_init` (const `vtss_inst_t` inst, const `vtss_ace_type_t` type, `vtss_ace_t` \*const ace)
 

*Initialize ACE to default values.*
- `vtss_rc vtss_ace_add` (const `vtss_inst_t` inst, const `vtss_ace_id_t` ace\_id\_next, const `vtss_ace_t` \*const ace)
 

*Add/modify ACE.*
- `vtss_rc vtss_ace_del` (const `vtss_inst_t` inst, const `vtss_ace_id_t` ace\_id)
 

*Delete ACE.*
- `vtss_rc vtss_ace_counter_get` (const `vtss_inst_t` inst, const `vtss_ace_id_t` ace\_id, `vtss_ace_counter_t` \*const counter)
 

*Get ACE counter.*
- `vtss_rc vtss_ace_counter_clear` (const `vtss_inst_t` inst, const `vtss_ace_id_t` ace\_id)
 

*Clear ACE counter.*

### 6.36.1 Detailed Description

Security API.

This header file describes security functions

### 6.36.2 Macro Definition Documentation

### 6.36.2.1 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.

## 6.36.3 Enumeration Type Documentation

### 6.36.3.1 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.

### 6.36.3.2 vtss\_acl\_port\_action\_t

```
enum vtss_acl_port_action_t
```

ACL port action.

Enumerator

VTSS_ACL_PORT_ACTION_NONE	No action from port list
VTSS_ACL_PORT_ACTION_FILTER	Port list filter is used
VTSS_ACL_PORT_ACTION_REDIR	Port list redirect is used

Definition at line 194 of file vtss\_security\_api.h.

### 6.36.3.3 vtss\_acl\_ptp\_action\_t

```
enum vtss_acl_ptp_action_t
```

ACL PTP action.

## Enumerator

VTSS_ACL_PTP_ACTION_NONE	No PTP action
VTSS_ACL_PTP_ACTION_ONE_STEP	PTP one-step time-stamping
VTSS_ACL_PTP_ACTION_ONE_STEP_ADD_DELAY	PTP one-step time-stamping, Serval: add delay, Jr2: Add EDLY
VTSS_ACL_PTP_ACTION_ONE_STEP_SUB_DELAY_1	PTP one-step time-stamping, Serval: subtract delay 1, Jr2: Add IDLY1
VTSS_ACL_PTP_ACTION_ONE_STEP_SUB_DELAY_2	PTP one-step time-stamping, Serval: subtract delay 2, Jr2: Add IDLY2
VTSS_ACL_PTP_ACTION_TWO_STEP	PTP two-step time-stamping

Definition at line 202 of file vtss\_security\_api.h.

6.36.3.4 `vtss_acl_ptp_rsp_t`

enum `vtss_acl_ptp_rsp_t`

ACL PTP response action.

## Enumerator

VTSS_ACL_PTP_RSP_NONE	No response action
VTSS_ACL_PTP_RSP_DLY_REQ_RSP_TS_UPD	Auto response to Delay_Req, includes receiveTimestamp update
VTSS_ACL_PTP_RSP_DLY_REQ_RSP_NO_TS	Auto response to Delay_Req, excludes receiveTimestamp update

Definition at line 220 of file vtss\_security\_api.h.

6.36.3.5 `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.

### 6.36.3.6 vtss\_ace\_bit\_t

enum `vtss_ace_bit_t`

ACE 1 bit.

Enumerator

<code>VTSS_ACE_BIT_ANY</code>	Value 0 or 1
<code>VTSS_ACE_BIT_0</code>	Value 0
<code>VTSS_ACE_BIT_1</code>	Value 1

Definition at line 355 of file vtss\_security\_api.h.

## 6.36.4 Function Documentation

### 6.36.4.1 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

<code>inst</code>	[IN] Target instance reference.
<code>port_no</code>	[IN] Port number.
<code>state</code>	[OUT] Authentication state.

Returns

Return code.

### 6.36.4.2 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.

### 6.36.4.3 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.

### 6.36.4.4 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.

**6.36.4.5 vtss\_acl\_port\_conf\_get()**

```
vtss_rc vtss_acl_port_conf_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const 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.

**6.36.4.6 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.

**6.36.4.7 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.

**6.36.4.8 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.

**6.36.4.9 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.

### 6.36.4.10 vtss\_ace\_add()

```
vtss_rc vtss_ace_add (
    const vtss_inst_t inst,
    const vtss_ace_id_t ace_id_next,
    const vtss_ace_t *const ace )
```

Add/modify ACE.

#### Parameters

<i>inst</i>	[IN] Target instance reference.
<i>ace_id_next</i>	[IN] ACE ID of next entry. The ACE will be added before the entry with this ID. VTSS_ACE_ID_LAST is reserved for inserting last.
<i>ace</i>	[IN] ACE structure.

#### Returns

Return code.

### 6.36.4.11 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.

Generated by Doxygen

**Returns**

Return code.

**6.36.4.12 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.

**6.36.4.13 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.

**6.37 vtss\_api/include/vtss\_sfi4\_api.h File Reference**

SFI4 API.

```
#include <vtss/api/types.h>
```

### 6.37.1 Detailed Description

SFI4 API.

## 6.38 vtss\_api/include/vtss\_sync\_api.h File Reference

Synchronization API.

```
#include "vtss/api/types.h"
```

### Data Structures

- struct `vtss_sync_clock_out_t`  
*Struct containing configuration for a recovered clock output port.*
- struct `vtss_sync_clock_in_t`  
*Struct containing configuration selecting the recovered input clock port, to be delivered to a selected output clock port.*

### Macros

- #define `VTSS_SYNC_CLK_A` 0
- #define `VTSS_SYNC_CLK_B` 1
- #define `VTSS_SYNC_CLK_MAX` 4

### TypeDefs

- typedef `u32 vtss_sync_clock_port_t`  
*Identification of a output clock port.*

### Enumerations

- enum `vtss_sync_divider_t`{ `VTSS_SYNC_DIVIDER_1`, `VTSS_SYNC_DIVIDER_4`, `VTSS_SYNC_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_clock_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_clock_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_clock_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_clock_port_t` clk\_port, `vtss_sync_clock_in_t` \*const conf)  
*Get the configuration of input port for a selected output clock port.*

### 6.38.1 Detailed Description

Synchronization API.

This header file describes synchronization functions

### 6.38.2 Macro Definition Documentation

#### 6.38.2.1 VTSS\_SYNCE\_CLK\_A

```
#define VTSS_SYNCE_CLK_A 0
```

Clock A output port

Definition at line 56 of file vtss\_sync\_api.h.

#### 6.38.2.2 VTSS\_SYNCE\_CLK\_B

```
#define VTSS_SYNCE_CLK_B 1
```

Clock B output port

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

#### 6.38.2.3 VTSS\_SYNCE\_CLK\_MAX

```
#define VTSS_SYNCE_CLK_MAX 4
```

Number of recovered clock outputs

Definition at line 59 of file vtss\_sync\_api.h.

### 6.38.3 Enumeration Type Documentation

#### 6.38.3.1 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.

#### 6.38.4 Function Documentation

##### 6.38.4.1 vtss\_sync\_clock\_out\_set()

```
vtss_rc vtss_sync_clock_out_set (
    const vtss_inst_t inst,
    const vtss_sync_clk_port_t clk_port,
    const vtss_sync_clock_out_t *const conf )
```

Set the configuration of a selected output clock port - against external clock controller.

###### Parameters

<i>inst</i>	[IN] handle to an API instance.
<i>clk_port</i>	[IN] Selection of the output clock port (VTSS_SYNCE_CLK_A or VTSS_SYNCE_CLK_B)
<i>conf</i>	[IN] pointer to a <a href="#">vtss_sync_clock_out_t</a> configuration structure.

###### Returns

Return code.

##### 6.38.4.2 vtss\_sync\_clock\_out\_get()

```
vtss_rc vtss_sync_clock_out_get (
    const vtss_inst_t inst,
    const vtss_sync_clk_port_t clk_port,
    vtss_sync_clock_out_t *const conf )
```

Get the configuration of a selected output clock port - against external clock controller.

###### Parameters

<i>inst</i>	[IN] handle to an API instance.
<i>clk_port</i>	[IN] Selection of the output clock port (VTSS_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.

**6.38.4.3 vtss\_sync\_clock\_in\_set()**

```
vtss_rc vtss_sync_clock_in_set (
    const vtss_inst_t inst,
    const vtss_sync_clk_port_t clk_port,
    const vtss_sync_clock_in_t *const conf )
```

Set the configuration of input port for a selected output clock port.

**Parameters**

<i>inst</i>	[IN] handle to an API instance.
<i>clk_port</i>	[IN] Selection of the output clock port (VTSS_SYNC_CLK_A or VTSS_SYNC_CLK_B)
<i>conf</i>	[IN] pointer to a <a href="#">vtss_sync_clock_in_t</a> configuration structure.

**Returns**

Return code.

**6.38.4.4 vtss\_sync\_clock\_in\_get()**

```
vtss_rc vtss_sync_clock_in_get (
    const vtss_inst_t inst,
    const vtss_sync_clk_port_t clk_port,
    vtss_sync_clock_in_t *const conf )
```

Get the configuration of input port for a selected output clock port.

**Parameters**

<i>inst</i>	[IN] handle to an API instance.
<i>clk_port</i>	[IN] Selection of the output clock port (VTSS_SYNC_CLK_A or VTSS_SYNC_CLK_B)
<i>conf</i>	[IN] pointer to a <a href="#">vtss_sync_clock_in_t</a> configuration structure.

**Returns**

Return code.

**6.39 vtss\_api/include/vtss\_tfi5\_api.h File Reference**

TFI5 API.

```
#include <vtss/api/types.h>
```

### 6.39.1 Detailed Description

TF15 API.

## 6.40 vtss\_api/include/vtss\_ts\_api.h File Reference

TimeStamping API.

```
#include <vtss/api/types.h>
```

### Data Structures

- struct [vtss\\_ts\\_alt\\_clock\\_mode\\_t](#)  
*parameter for setting the alternative clock mode.*
- 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](#) 0 /\* time counter wrap around limit+1 \*/  
*Jaguar2 nanosecond time counter wrap around value (jaguar2 time counter wraps when 0xffffffff is reached).*
- #define [VTSS\\_HW\\_TIME\\_MIN\\_ADJ\\_RATE](#) 10 /\* 1 ppb \*/  
*Jaguar 2 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_alt_clock_mode_t vtss_ts_alt_clock_mode_t`  
*parameter for setting the alternative clock mode.*
- `typedef struct vtss_ts_ext_clock_mode_t vtss_ts_ext_clock_mode_t`  
*external clock output configuration.*
- `typedef struct vtss_ts_operation_mode_t vtss_ts_operation_mode_t`  
*Timestamp operation.*
- `typedef struct vtss_ts_internal_mode_t vtss_ts_internal_mode_t`  
*Hardware timestamping format mode for internal ports.*
- `typedef struct vtss_ts_id_t vtss_ts_id_t`  
*Timestamp identifier.*
- `typedef struct vtss_ts_timestamp_t vtss_ts_timestamp_t`  
*Timestamp structure.*
- `typedef struct vtss_ts_timestamp_alloc_t vtss_ts_timestamp_alloc_t`  
*Timestamp allocation.*

## Enumerations

- `enum vtss_ts_ext_clock_one_pps_mode_t { TS_EXT_CLOCK_MODE_ONE_PPS_DISABLE, TS_EXT_CLOCK_MODE_ONE_PPS_OUTPUT, TS_EXT_CLOCK_MODE_ONE_PPS_INPUT, TS_EXT_CLOCK_MODE_ONE_PPS_OUTPUT_INPUT, TS_EXT_CLOCK_MODE_MAX }`  
*parameter for setting the external clock mode.*
- `enum vtss_ts_mode_t { TS_MODE_NONE, TS_MODE_EXTERNAL, TS_MODE_INTERNAL, TX_MODE_MAX }`  
*parameter for setting the timestamp operating mode*
- `enum vtss_ts_internal_fmt_t { TS_INTERNAL_FMT_NONE, TS_INTERNAL_FMT_RESERVED_LEN_30BIT, TS_INTERNAL_FMT_RESERVED_LEN_32BIT, TS_INTERNAL_FMT_SUB_ADD_LEN_44BIT_CF62, TS_INTERNAL_FMT_RESERVED_LEN_48BIT_CF_3_0, TS_INTERNAL_FMT_RESERVED_LEN_48BIT_CF_0, TS_INTERNAL_FMT_RESERVED_LEN_48BIT_CF, TS_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_alt_clock_saved_get (const vtss_inst_t inst, u32 *const saved)`  
*Get the latest saved nanosec counter from the alternative clock.*
- `vtss_rc vtss_ts_alt_clock_mode_get (const vtss_inst_t inst, vtss_ts_alt_clock_mode_t *const alt_clock_mode)`  
*Get the alternative external clock mode.*
- `vtss_rc vtss_ts_alt_clock_mode_set (const vtss_inst_t inst, const vtss_ts_alt_clock_mode_t *const alt_clock_mode)`  
*Set the alternative external clock mode. This function configures the 1PPS, L/S pin usage for pin set no 0 in Serval.*
- `vtss_rc vtss_ts_timeofday_next_pps_set (const vtss_inst_t inst, const vtss_timestamp_t *const ts)`  
*Set the time at the next 1PPS pulse edge in a Timestamp format.*
- `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)*

#### 6.40.1 Detailed Description

TimeStamping API.

This header file describes PTP/OAM TimeStamping API functions and associated types.

#### 6.40.2 Typedef Documentation

##### 6.40.2.1 `vtss_ts_alt_clock_mode_t`

```
typedef struct vtss_ts_alt_clock_mode_t vtss_ts_alt_clock_mode_t
```

parameter for setting the alternative clock mode.

external clock output configuration.

#### 6.40.3 Function Documentation

##### 6.40.3.1 `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.

**6.40.3.2 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.

**6.40.3.3 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.

#### 6.40.3.4 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

#### 6.40.3.5 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.

#### 6.40.3.6 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.

**6.40.3.7 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.

**6.40.3.8 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.

**6.40.3.9 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.

**6.40.3.10 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.

**6.40.3.11 vtss\_ts\_alt\_clock\_saved\_get()**

```
vtss_rc vtss_ts_alt_clock_saved_get (
    const vtss_inst_t inst,
    u32 *const saved )
```

Get the latest saved nanosec counter from the alternative clock.

**Parameters**

<i>inst</i>	[IN] handle to an API instance
<i>saved</i>	[OUT] latest saved value.

**Returns**

Return code.

**6.40.3.12 vtss\_ts\_alt\_clock\_mode\_get()**

```
vtss_rc vtss_ts_alt_clock_mode_get (
    const vtss_inst_t inst,
    vtss_ts_alt_clock_mode_t *const alt_clock_mode )
```

Get the alternative external clock mode.

**Parameters**

<i>inst</i>	[IN] handle to an API instance
<i>alt_clock_mode</i>	[OUT] alternative clock mode.

**Returns**

Return code.

**6.40.3.13 vtss\_ts\_alt\_clock\_mode\_set()**

```
vtss_rc vtss_ts_alt_clock_mode_set (
    const vtss_inst_t inst,
    const vtss_ts_alt_clock_mode_t *const alt_clock_mode )
```

Set the alternative external clock mode. This function configures the 1PPS, L/S pin usage for pin set no 0 in Serval.

**Parameters**

<i>inst</i>	[IN] handle to an API instance
<i>alt_clock_mode</i>	[IN] alternative clock mode.

**Returns**

Return code.

#### 6.40.3.14 vtss\_ts\_timeofday\_next\_pps\_set()

```
vtss_rc vtss_ts_timeofday_next_pps_set (
    const vtss_inst_t inst,
    const vtss_timestamp_t *const ts )
```

Set 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.

#### 6.40.3.15 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.

#### 6.40.3.16 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.

**6.40.3.17 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.

**6.40.3.18 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.

### 6.40.3.19 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.

### 6.40.3.20 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.

### 6.40.3.21 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.

### 6.40.3.22 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.

### 6.40.3.23 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.

**6.40.3.24 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.

**6.40.3.25 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.

#### 6.40.3.26 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

#### 6.40.3.27 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.

#### 6.40.3.28 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

**6.40.3.29 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.

**6.40.3.30 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.

#### 6.40.3.31 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<sub>←</sub> _id</i>	[IN] timestamp id
<i>ts</i>	[OUT] pointer to a struct holding the fifo timestamp

##### Returns

Return code.

#### 6.40.3.32 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<sub>←</sub> _id</i>	[IN] timestamp id

##### Returns

Return code.

#### 6.40.3.33 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.

**6.40.3.34 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.

**6.40.3.35 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.

#### 6.40.3.36 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.

## 6.41 vtss\_api/include/vtss\_upi\_api.h File Reference

Define UPI API interface.

```
#include <vtss/api/types.h>
```

### 6.41.1 Detailed Description

Define UPI API interface.

## 6.42 vtss\_api/include/vtss\_wis\_api.h File Reference

eWIS layer API

```
#include <vtss/api/types.h>
```

## Data Structures

- struct `vtss_ewis_tti_s`  
*Trail Trace Identifier type.*
- struct `vtss_ewis_fault_cons_act_s`  
*eWIS fault mask configuration, i.e set up which defects trigger the Fault condition*
- struct `vtss_ewis_aisl_cons_act_s`  
*eWIS AIS-L consequent actions*
- struct `vtss_ewis_rdil_cons_act_s`  
*eWIS RDI-L consequent actions*
- struct `vtss_ewis_cons_act_s`

- 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 }**  
*Trail 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)**

- **`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)
 

*Polling function called at by interrupt or periodically.*
- **`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)
 

*Forces one or more WIS events to occur (simulated events)*
- **`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_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. \**

- `vtss_rc vtss_ewis_path_txti_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_ewis_txti_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 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_txti_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_txti_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.*

### 6.42.1 Detailed Description

eWIS layer API

### 6.42.2 Macro Definition Documentation

#### 6.42.2.1 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`.

#### 6.42.2.2 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`.

#### 6.42.2.3 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`.

#### 6.42.2.4 VTSS\_EWIS\_LOF\_EV

```
#define VTSS_EWIS_LOF_EV 0x00000008
```

Loss of Frame (LOF)

Definition at line 341 of file vtss\_wis\_api.h.

#### 6.42.2.5 VTSS\_EWIS\_LOS\_EV

```
#define VTSS_EWIS_LOS_EV 0x000000010
```

Loss of Signal (LOS)

Definition at line 342 of file vtss\_wis\_api.h.

#### 6.42.2.6 VTSS\_EWIS\_RDIL\_EV

```
#define VTSS_EWIS_RDIL_EV 0x000000020
```

Line Remote Defect Indication (RDI-L)

Definition at line 343 of file vtss\_wis\_api.h.

#### 6.42.2.7 VTSS\_EWIS\_AISL\_EV

```
#define VTSS_EWIS_AISL_EV 0x000000040
```

Line Alarm Indication Signal (AIS-L)

Definition at line 344 of file vtss\_wis\_api.h.

#### 6.42.2.8 VTSS\_EWIS\_LCDP\_EV

```
#define VTSS_EWIS_LCDP_EV 0x000000080
```

Loss of Code-group Delineation (LCD-P)

Definition at line 345 of file vtss\_wis\_api.h.

#### 6.42.2.9 VTSS\_EWIS\_PLMP\_EV

```
#define VTSS_EWIS_PLMP_EV 0x00000100
```

Path Label Mismatch (PLMP)

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

#### 6.42.2.10 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.

#### 6.42.2.11 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.

#### 6.42.2.12 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.

#### 6.42.2.13 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.

**6.42.2.14 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.

**6.42.2.15 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.

**6.42.2.16 VTSS\_EWIS\_UNEQP\_EV**

```
#define VTSS_EWIS_UNEQP_EV 0x00008000
```

Unequiped Path (UNEQ-P)

Definition at line 353 of file vtss\_wis\_api.h.

**6.42.2.17 VTSS\_EWIS\_FEUNEQP\_EV**

```
#define VTSS_EWIS_FEUNEQP_EV 0x00010000
```

Far-end Unequiped Path (UNEQ-P)

Definition at line 354 of file vtss\_wis\_api.h.

**6.42.2.18 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.

**6.42.2.19 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.

**6.42.2.20 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.

**6.42.2.21 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.

**6.42.2.22 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.

**6.42.2.23 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.

**6.42.2.24 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.

**6.42.2.25 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.

**6.42.2.26 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.

**6.42.2.27 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.

**6.42.2.28 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.

#### 6.42.2.29 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.

#### 6.42.2.30 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.

#### 6.42.2.31 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.

#### 6.42.2.32 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.

### 6.42.3 Typedef Documentation

#### 6.42.3.1 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.

### 6.42.3.2 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.

## 6.42.4 Enumeration Type Documentation

### 6.42.4.1 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.

### 6.42.4.2 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.

### 6.42.4.3 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.

## 6.42.4.4 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.

## 6.42.4.5 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.

## 6.42.5 Function Documentation

#### 6.42.5.1 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.

#### 6.42.5.2 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.

#### 6.42.5.3 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.

#### 6.42.5.4 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.

**6.42.5.5 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.

**6.42.5.6 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.

#### 6.42.5.7 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.

#### 6.42.5.8 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.

#### 6.42.5.9 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.

**6.42.5.10 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.

**6.42.5.11 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.

**6.42.5.12 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.

**6.42.5.13 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.

**6.42.5.14 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.

**6.42.5.15 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.

#### 6.42.5.16 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.

#### 6.42.5.17 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_ttxi_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 ttxi.

##### Returns

Return code.

#### 6.42.5.18 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_ttxi_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>txti</i>	[OUT] pointer to transmitted tti.

**Returns**

Return code.

**6.42.5.19 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.

**6.42.5.20 vtss\_ewis\_path\_txci\_set()**

```
vtss_rc vtss_ewis_path_txci_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_ewis_txci_t *const txci )
```

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>txci</i>	[IN] pointer to transmitted txci.

**Returns**

Return code.

**6.42.5.21 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_txti_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.

**6.42.5.22 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)

#### 6.42.5.23 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.

#### 6.42.5.24 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)

**6.42.5.25 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.

**6.42.5.26 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.

**6.42.5.27 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.

**6.42.5.28 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.

#### 6.42.5.29 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.

#### 6.42.5.30 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.

#### 6.42.5.31 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.

### 6.42.5.32 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.

### 6.42.5.33 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.

**6.42.5.34 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.

**6.42.5.35 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.

## 6.43 vtss\_api/include/vtss\_xaui\_api.h File Reference

XAUI API.

```
#include <vtss/api/types.h>
```

### 6.43.1 Detailed Description

XAUI API.

## 6.44 vtss\_api/include/vtss\_xfi\_api.h File Reference

XFI API.

```
#include <vtss/api/types.h>
```

### 6.44.1 Detailed Description

XFI API.



# Index

a\_gpio  
    vtss\_phy\_10g\_auto\_failover\_conf\_t, 209

AMPLITUDE\_POINTS  
    vtss\_phy\_10g\_api.h, 798

ach\_opt  
    vtss\_phy\_ts\_oam\_engine\_flow\_conf\_t, 374  
    vtss\_phy\_ts\_ptp\_engine\_flow\_conf\_t, 381

acknowledge  
    vtss\_phy\_10g\_clause\_37\_adv\_t, 223  
    vtss\_port\_clause\_37\_adv\_t, 398

acl\_hit  
    vtss\_packet\_rx\_info\_t, 178

acl\_idx  
    vtss\_packet\_rx\_info\_t, 178

act\_len  
    tag\_vtss\_fdma\_list, 31

action  
    vtss\_ace\_t, 53  
    vtss\_acl\_port\_conf\_t, 61  
    vtss\_phy\_ts\_engine\_action\_t, 339  
    vtss\_qce\_t, 446  
    vtss\_vce\_t, 510

action\_gen  
    vtss\_phy\_ts\_engine\_action\_t, 339

action\_ptp  
    vtss\_phy\_ts\_engine\_action\_t, 339

active  
    vtss\_irq\_status\_t, 149  
    vtss\_phy\_10g\_kr\_status\_aneg\_t, 243  
    vtss\_phy\_10g\_prbs\_mon\_conf\_t, 280

addr  
    vtss\_ip\_addr\_t, 142  
    vtss\_ipv6\_t, 146  
    vtss\_mac\_t, 155  
    vtss\_phy\_ts\_ip\_conf\_t, 364  
    vtss\_wol\_mac\_addr\_t, 530

addr\_match\_mode  
    vtss\_phy\_ts\_eth\_conf\_t, 344

addr\_match\_select  
    vtss\_phy\_ts\_eth\_conf\_t, 344

address  
    vtss\_ip\_network\_t, 142  
    vtss\_ipv4\_network\_t, 143  
    vtss\_ipv6\_network\_t, 145

adv\_10g  
    vtss\_phy\_10g\_base\_kr\_id\_adv\_abil\_t, 214

adv\_1g  
    vtss\_phy\_10g\_base\_kr\_id\_adv\_abil\_t, 214

adv\_dis

port\_custom\_conf\_t, 28

advertisement  
    vtss\_phy\_10g\_clause\_37\_control\_t, 225  
    vtss\_port\_clause\_37\_control\_t, 400

agc  
    vtss\_phy\_10g\_ib\_conf\_t, 234

aged  
    vtss\_mac\_table\_entry\_t, 157  
    vtss\_mac\_table\_status\_t, 160

aggr\_code  
    vtss\_packet\_tx\_info\_t, 198

aggr\_intrpt  
    vtss\_gpio\_10g\_gpio\_mode\_t, 134

aggr\_rx\_disable  
    vtss\_packet\_frame\_info\_t, 167  
    vtss\_packet\_port\_info\_t, 169

aggr\_tx\_disable  
    vtss\_packet\_frame\_info\_t, 167  
    vtss\_packet\_port\_info\_t, 170

ais\_on\_lof  
    vtss\_ewis\_aisl\_cons\_act\_s, 80

ais\_on\_los  
    vtss\_ewis\_aisl\_cons\_act\_s, 80

aisl  
    vtss\_ewis\_cons\_act\_s, 85

alloc\_ptr  
    tag\_vtss\_fdma\_list, 32

amp\_range  
    vtss\_phy\_10g\_vscope\_scan\_status\_t, 300

ampl  
    vtss\_phy\_10g\_base\_kr\_conf\_t, 213

an\_enable  
    vtss\_phy\_10g\_base\_kr\_autoneg\_t, 211

an\_reset  
    vtss\_phy\_10g\_base\_kr\_autoneg\_t, 211

an\_restart  
    vtss\_phy\_10g\_base\_kr\_autoneg\_t, 211

ana\_sync  
    vtss\_ewis\_test\_status\_s, 111

aneg  
    vtss\_phy\_10g\_base\_kr\_status\_t, 215  
    vtss\_phy\_conf\_t, 306  
    vtss\_port\_status\_t, 431

aneg\_complete  
    vtss\_port\_sgmii\_aneg\_t, 429  
    vtss\_port\_status\_t, 430

aneg\_enable  
    vtss\_phy\_tbi\_conf\_t, 332

aneg\_pd\_detect

vtss\_phy\_media\_serd\_pcs\_cntl\_t, 321  
 aneg\_restart  
     vtss\_phy\_mac\_serd\_pcs\_cntl\_t, 318  
 apc\_bit\_mask  
     vtss\_phy\_10g\_ib\_conf\_t, 236  
 apc\_eqz\_offs\_par\_cfg  
     vtss\_phy\_10g\_mode\_t, 258  
 apc\_host\_eqz\_ld\_ctrl  
     vtss\_phy\_10g\_mode\_t, 258  
 apc\_host\_ld\_ctrl  
     vtss\_phy\_10g\_mode\_t, 257  
 apc\_ib\_regulator  
     vtss\_phy\_10g\_mode\_t, 259  
 apc\_ib\_regulator\_t  
     vtss\_phy\_10g\_api.h, 815  
 apc\_line\_eqz\_ld\_ctrl  
     vtss\_phy\_10g\_mode\_t, 258  
 apc\_line\_ld\_ctrl  
     vtss\_phy\_10g\_mode\_t, 257  
 apc\_offs\_ctrl  
     vtss\_phy\_10g\_mode\_t, 256  
 arp  
     vtss\_ace\_frame\_arp\_t, 34  
     vtss\_ace\_t, 54  
 arrived\_tagged  
     vtss\_packet\_rx\_header\_t, 173  
 asymmetric\_pause  
     vtss\_phy\_10g\_clause\_37\_adv\_t, 222  
     vtss\_phy\_aneg\_t, 302  
     vtss\_port\_clause\_37\_adv\_t, 398  
 auto\_clear\_ls  
     vtss\_phy\_ts\_init\_conf\_t, 360  
 automatic  
     vtss\_learn\_mode\_t, 154  
 autoneg  
     port\_custom\_conf\_t, 27  
     vtss\_phy\_10g\_base\_kr\_train\_aneg\_t, 217  
     vtss\_phy\_10g\_clause\_37\_status\_t, 227  
 BOOLEAN\_STORAGE\_COUNT  
     vtss\_phy\_10g\_api.h, 798  
 BOOL  
     types.h, 596  
 bad\_crc  
     vtss\_phy\_10g\_pkt\_mon\_conf\_t, 272  
 base\_port\_no  
     vtss\_phy\_type\_t, 388  
 ber  
     vtss\_phy\_10g\_pkt\_mon\_conf\_t, 273  
     vtss\_phy\_10g\_vscope\_scan\_conf\_t, 299  
 ber\_cnt  
     vtss\_phy\_pcs\_cnt\_t, 323  
 bin  
     vtss\_packet\_tx\_info\_t, 200  
 bist\_mode  
     vtss\_phy\_10g\_prbs\_mon\_conf\_t, 279  
 bit\_count  
     vtss\_sgpio\_conf\_t, 468  
 bit\_errors  
     vtss\_phy\_10g\_ib\_status\_t, 237  
 bit\_rate  
     vtss\_acl\_policer\_conf\_t, 60  
 bit\_rate\_enable  
     vtss\_acl\_policer\_conf\_t, 60  
 block\_lock  
     vtss\_phy\_10g\_kr\_status\_aneg\_t, 245  
     vtss\_phy\_10g\_status\_t, 294  
 block\_lock\_latched  
     vtss\_phy\_pcs\_cnt\_t, 323  
 bmode  
     vtss\_sgpio\_conf\_t, 467  
 bottom\_up  
     vtss\_phy\_ts\_mpls\_conf\_t, 368  
 bpdu\_cpu\_only  
     vtss\_packet\_rx\_reg\_t, 193  
 bpdu\_queue  
     vtss\_packet\_rx\_queue\_map\_t, 190  
 bpdu\_reg  
     vtss\_packet\_rx\_port\_conf\_t, 188  
 bridge  
     vtss\_port\_counters\_t, 405  
 bypass\_in\_api  
     vtss\_phy\_10g\_fifo\_sync\_t, 229  
 byte\_limit\_per\_tick  
     vtss\_fdma\_throttle\_cfg\_t, 131  
 c  
     vtss\_phy\_10g\_ib\_conf\_t, 234  
 c0  
     vtss\_phy\_10g\_base\_kr\_conf\_t, 212  
 c0\_ob\_tap\_result  
     vtss\_phy\_10g\_kr\_status\_train\_t, 247  
 c1  
     vtss\_phy\_10g\_base\_kr\_conf\_t, 212  
 c\_ctrl  
     vtss\_phy\_10g\_ob\_status\_t, 264  
 c\_intrpt  
     vtss\_gpio\_10g\_gpio\_mode\_t, 134  
 CALIB\_DONE  
     vtss\_phy\_ts\_nphase\_status\_t, 371  
 CALIB\_ERR  
     vtss\_phy\_ts\_nphase\_status\_t, 370  
 CHIP\_PORT\_UNUSED  
     vtss\_port\_api.h, 1024  
 cal\_done  
     vtss\_lcpll\_status\_t, 153  
 cal\_error  
     vtss\_lcpll\_status\_t, 153  
     vtss\_rcpll\_status\_t, 459  
 cal\_not\_done  
     vtss\_rcpll\_status\_t, 459  
 cb  
     vtss\_ts\_timestamp\_alloc\_t, 485  
 cbs  
     vtss\_dlb\_policer\_conf\_t, 74  
 cf  
     vtss\_dlb\_policer\_conf\_t, 73  
 cf\_update

vtss\_phy\_ts\_ptp\_engine\_action\_t, 379  
cfg  
  vtss\_phy\_conf\_1g\_t, 305  
cfg0  
  vtss\_phy\_10g\_mode\_t, 257  
cfi  
  vtss\_ace\_vlan\_t, 56  
  vtss\_tci\_t, 476  
channel\_conf  
  vtss\_phy\_10g\_init\_parm\_t, 241  
channel\_id  
  vtss\_phy\_10g\_auto\_failover\_conf\_t, 209  
  vtss\_phy\_10g\_id\_t, 240  
  vtss\_phy\_10g\_mode\_t, 254  
  vtss\_phy\_type\_t, 388  
channel\_map  
  vtss\_phy\_ts\_engine\_flow\_conf\_t, 340  
  vtss\_phy\_ts\_generic\_action\_t, 354  
  vtss\_phy\_ts\_oam\_engine\_action\_t, 372  
  vtss\_phy\_ts\_ptp\_engine\_action\_t, 378  
channel\_type  
  vtss\_phy\_ts\_ach\_conf\_t, 335  
chip\_no  
  vtss\_debug\_info\_t, 70  
  vtss\_debug\_lock\_t, 71  
  vtss\_fdma\_ch\_cfg\_t, 129  
  vtss\_packet\_rx\_meta\_t, 184  
  vtss\_port\_map\_t, 417  
chip\_port  
  vtss\_port\_map\_t, 417  
  vtss\_vstax\_tx\_header\_t, 529  
chk\_1ng\_modified  
  vtss\_phy\_ts\_init\_conf\_t, 361  
cir  
  vtss\_dlb\_policer\_conf\_t, 74  
ckout\_sel  
  vtss\_phy\_10g\_ckout\_conf\_t, 221  
ckout\_sel\_  
  vtss\_phy\_10g\_api.h, 822  
ckout\_sel\_t  
  vtss\_phy\_10g\_api.h, 811  
clear  
  vtss\_debug\_info\_t, 70  
clk\_freq  
  vtss\_phy\_ts\_init\_conf\_t, 358  
clk\_mode  
  vtss\_phy\_ts\_ptp\_engine\_action\_t, 379  
clk\_mstr\_t  
  vtss\_phy\_10g\_api.h, 816  
clk\_sel\_no  
  vtss\_phy\_10g\_host\_clk\_conf\_t, 232  
  vtss\_phy\_10g\_line\_clk\_conf\_t, 250  
clk\_src  
  vtss\_phy\_ts\_init\_conf\_t, 358  
cm  
  vtss\_dlb\_policer\_conf\_t, 73  
cm1  
  vtss\_phy\_10g\_base\_kr\_conf\_t, 212  
cm\_ob\_tap\_result  
  vtss\_phy\_10g\_kr\_status\_train\_t, 247  
cmef\_disable  
  vtss\_vstax\_conf\_t, 521  
comm\_opt  
  vtss\_phy\_ts\_ach\_conf\_t, 335  
  vtss\_phy\_ts\_eth\_conf\_t, 343  
  vtss\_phy\_ts\_gen\_conf\_t, 352  
  vtss\_phy\_ts\_ip\_conf\_t, 363  
  vtss\_phy\_ts\_mpls\_conf\_t, 366  
complete  
  vtss\_phy\_10g\_clause\_37\_status\_t, 226  
  vtss\_phy\_10g\_kr\_status\_aneg\_t, 243  
  vtss\_phy\_10g\_kr\_status\_train\_t, 247  
config\_bit\_mask  
  vtss\_phy\_10g\_ib\_conf\_t, 236  
connector\_enable  
  vtss\_phy\_loopback\_t, 314  
context  
  vtss\_ts\_timestamp\_alloc\_t, 484  
  vtss\_ts\_timestamp\_t, 486  
copper  
  vtss\_port\_status\_t, 431  
copy\_smac\_to\_dmac  
  vtss\_acl\_ptp\_action\_conf\_t, 62  
copy\_to\_cpu  
  vtss\_mac\_table\_entry\_t, 157  
corrected\_block\_cnt  
  vtss\_phy\_10g\_kr\_status\_fec\_t, 246  
cos  
  vtss\_packet\_rx\_info\_t, 178  
  vtss\_packet\_tx\_info\_t, 198  
cp\_ob\_tap\_result  
  vtss\_phy\_10g\_kr\_status\_train\_t, 247  
cpu  
  vtss\_acl\_action\_t, 57  
  vtss\_learn\_mode\_t, 154  
cpu\_once  
  vtss\_acl\_action\_t, 57  
cpu\_queue  
  vtss\_acl\_action\_t, 57  
  vtss\_mac\_table\_entry\_t, 157  
  vtss\_policer\_ext\_t, 395  
cs\_wait\_ns  
  vtss\_pi\_conf\_t, 392  
cu\_bad  
  vtss\_phy\_statistic\_t, 329  
cu\_good  
  vtss\_phy\_statistic\_t, 329  
cur\_version  
  vtss\_restart\_status\_t, 462  
cw\_en  
  vtss\_phy\_ts\_mpls\_conf\_t, 366  
d\_filter  
  vtss\_phy\_10g\_mode\_t, 257  
d\_filt  
  vtss\_phy\_10g\_ob\_status\_t, 265  
dais\_l

vtss\_ewis\_defects\_s, 90  
 dais\_p  
     vtss\_ewis\_defects\_s, 90  
 data  
     vtss\_ace\_frame\_etype\_t, 37  
     vtss\_ace\_frame\_ipv4\_t, 40  
     vtss\_ace\_frame\_ipv6\_t, 44  
     vtss\_phy\_ts\_gen\_conf\_t, 353  
     vtss\_phy\_ts\_generic\_action\_t, 354  
     vtss\_qce\_frame\_etype\_t, 435  
     vtss\_qce\_frame\_llc\_t, 440  
     vtss\_qce\_frame\_snap\_t, 440  
     vtss\_vce\_frame\_etype\_t, 500  
     vtss\_vce\_frame\_llc\_t, 504  
     vtss\_vce\_frame\_snap\_t, 505  
 ddr\_mode  
     vtss\_phy\_10g\_mode\_t, 260  
 ddr\_mode\_t  
     vtss\_phy\_10g\_api.h, 815  
 default\_dei  
     vtss\_qos\_port\_conf\_t, 455  
 default\_dpl  
     vtss\_qos\_port\_conf\_t, 455  
 default\_prio  
     vtss\_qos\_port\_conf\_t, 454  
 dei  
     vtss\_mirror\_conf\_t, 162  
     vtss\_qce\_action\_t, 434  
     vtss\_qce\_tag\_t, 447  
     vtss\_vce\_tag\_t, 511  
     vtss\_vlan\_tag\_t, 516  
 delaym\_type  
     vtss\_phy\_ts\_ietf\_mpls\_ach\_oam\_conf\_t, 357  
     vtss\_phy\_ts\_ptp\_engine\_action\_t, 379  
     vtss\_phy\_ts\_y1731\_oam\_conf\_t, 385  
 des\_interface\_width  
     vtss\_phy\_10g\_prbs\_mon\_conf\_t, 278  
 dest\_ip  
     vtss\_phy\_ts\_fifo\_sig\_t, 351  
 dest\_mac  
     vtss\_phy\_ts\_fifo\_sig\_t, 351  
 destination  
     vtss\_ipv4\_uc\_t, 144  
     vtss\_ipv6\_uc\_t, 147  
     vtss\_irq\_conf\_t, 148  
     vtss\_mac\_table\_entry\_t, 157  
 detect\_only  
     vtss\_phy\_ts\_fifo\_conf\_t, 348  
 device\_feature\_status  
     vtss\_phy\_10g\_id\_t, 240  
 dfais\_p  
     vtss\_ewis\_defects\_s, 91  
 dfe1  
     vtss\_phy\_10g\_ib\_conf\_t, 234  
 dfe2  
     vtss\_phy\_10g\_ib\_conf\_t, 234  
 dfe3  
     vtss\_phy\_10g\_ib\_conf\_t, 235  
 dfe4  
     vtss\_phy\_10g\_ib\_conf\_t, 235  
 dfplm\_p  
     vtss\_ewis\_defects\_s, 91  
 dfuneq\_p  
     vtss\_ewis\_defects\_s, 92  
 dgroup\_no  
     vtss\_dgroup\_port\_conf\_t, 72  
 dig\_offset\_reg  
     vtss\_phy\_10g\_mode\_t, 256  
 dip  
     vtss\_ace\_frame\_arp\_t, 36  
     vtss\_ace\_frame\_ipv4\_t, 39  
     vtss\_qce\_frame\_ipv4\_t, 437  
     vtss\_qce\_frame\_ipv6\_t, 438  
 disable  
     vtss\_phy\_mac\_serdes\_pcs\_cntl\_t, 317  
 discard  
     vtss\_learn\_mode\_t, 155  
 divider  
     vtss\_sync\_clock\_out\_t, 474  
 dlcd\_p  
     vtss\_ewis\_defects\_s, 91  
 dllof\_s  
     vtss\_ewis\_defects\_s, 89  
 dllop\_p  
     vtss\_ewis\_defects\_s, 90  
 dlos\_s  
     vtss\_ewis\_defects\_s, 89  
 dma\_enable  
     vtss\_packet\_dma\_conf\_t, 165  
 dmac  
     vtss\_ace\_frame\_etype\_t, 36  
     vtss\_ace\_frame\_llc\_t, 47  
     vtss\_ace\_frame\_snap\_t, 48  
     vtss\_phy\_10g\_pkt\_gen\_conf\_t, 270  
     vtss\_qce\_mac\_t, 444  
 dmac\_bc  
     vtss\_ace\_t, 53  
     vtss\_qce\_mac\_t, 444  
     vtss\_vce\_mac\_t, 508  
 dmac\_dip  
     vtss\_vcl\_port\_conf\_t, 512  
 dmac\_enable  
     vtss\_aggr\_mode\_t, 64  
 dmac\_match  
     vtss\_ace\_frame\_arp\_t, 35  
 dmac\_mc  
     vtss\_ace\_t, 53  
     vtss\_qce\_mac\_t, 444  
     vtss\_vce\_mac\_t, 508  
 dom\_sel  
     vtss\_acl\_ptp\_action\_conf\_t, 63  
 domain  
     vtss\_phy\_ts\_ptp\_conf\_t, 377  
 domain\_num  
     vtss\_phy\_ts\_fifo\_sig\_t, 350  
 doof\_s

vtss\_ewis\_defects\_s, 89  
dot1dTpPortInDiscards  
    vtss\_port\_bridge\_counters\_t, 397  
dot3ControlInUnknownOpcodes  
    vtss\_port\_ethernet\_like\_counters\_t, 407  
dot3InPauseFrames  
    vtss\_port\_ethernet\_like\_counters\_t, 407  
dot3OutPauseFrames  
    vtss\_port\_ethernet\_like\_counters\_t, 409  
dot3StatsAlignmentErrors  
    vtss\_port\_ethernet\_like\_counters\_t, 406  
dot3StatsCarrierSenseErrors  
    vtss\_port\_ethernet\_like\_counters\_t, 409  
dot3StatsDeferredTransmissions  
    vtss\_port\_ethernet\_like\_counters\_t, 408  
dot3StatsExcessiveCollisions  
    vtss\_port\_ethernet\_like\_counters\_t, 408  
dot3StatsFCSErrors  
    vtss\_port\_ethernet\_like\_counters\_t, 407  
dot3StatsFrameTooLongs  
    vtss\_port\_ethernet\_like\_counters\_t, 407  
dot3StatsLateCollisions  
    vtss\_port\_ethernet\_like\_counters\_t, 408  
dot3StatsMultipleCollisionFrames  
    vtss\_port\_ethernet\_like\_counters\_t, 408  
dot3StatsSingleCollisionFrames  
    vtss\_port\_ethernet\_like\_counters\_t, 408  
dot3StatsSymbolErrors  
    vtss\_port\_ethernet\_like\_counters\_t, 407  
dp  
    vtss\_packet\_tx\_info\_t, 204  
    vtss\_qce\_action\_t, 433  
    vtss\_vstax\_tx\_header\_t, 529  
dp\_bypass\_level  
    vtss\_policer\_ext\_t, 393  
dp\_enable  
    vtss\_qce\_action\_t, 433  
dp\_level\_map  
    vtss\_qos\_port\_conf\_t, 455  
dpIm\_p  
    vtss\_ewis\_defects\_s, 91  
dport  
    vtss\_ace\_frame\_ipv4\_t, 40  
    vtss\_ace\_frame\_ipv6\_t, 44  
    vtss\_qce\_frame\_ipv4\_t, 437  
    vtss\_qce\_frame\_ipv6\_t, 439  
    vtss\_vce\_frame\_ipv4\_t, 502  
    vtss\_vce\_frame\_ipv6\_t, 503  
dport\_mask  
    vtss\_phy\_ts\_ip\_conf\_t, 363  
dport\_val  
    vtss\_phy\_ts\_ip\_conf\_t, 363  
drdi\_l  
    vtss\_ewis\_defects\_s, 90  
drdi\_p  
    vtss\_ewis\_defects\_s, 91  
ds  
    vtss\_ace\_frame\_ipv4\_t, 39  
vtss\_ace\_frame\_ipv6\_t, 44  
vtss\_phy\_ts\_ietf\_mpls\_ach\_oam\_conf\_t, 357  
dscp  
    vtss\_qce\_action\_t, 433  
    vtss\_qce\_frame\_ipv4\_t, 436  
    vtss\_qce\_frame\_ipv6\_t, 438  
    vtss\_vce\_frame\_ipv4\_t, 501  
    vtss\_vce\_frame\_ipv6\_t, 503  
dscp\_class\_enable  
    vtss\_qos\_port\_conf\_t, 456  
dscp\_dp\_level\_map  
    vtss\_qos\_conf\_t, 449  
dscp\_emode  
    vtss\_qos\_port\_conf\_t, 456  
dscp\_enable  
    vtss\_qce\_action\_t, 433  
dscp\_mode  
    vtss\_qos\_port\_conf\_t, 456  
dscp\_qos\_class\_map  
    vtss\_qos\_conf\_t, 449  
dscp\_qos\_map  
    vtss\_qos\_conf\_t, 449  
dscp\_qos\_map\_dp1  
    vtss\_qos\_conf\_t, 449  
dscp\_qos\_map\_dp2  
    vtss\_qos\_conf\_t, 449  
dscp\_qos\_map\_dp3  
    vtss\_qos\_conf\_t, 450  
dscp\_remap  
    vtss\_qos\_conf\_t, 450  
dscp\_remark  
    vtss\_qos\_conf\_t, 450  
dscp\_translate  
    vtss\_qos\_port\_conf\_t, 456  
dscp\_translate\_map  
    vtss\_qos\_conf\_t, 450  
dscp\_trust  
    vtss\_qos\_conf\_t, 448  
dst\_port\_mask  
    vtss\_packet\_tx\_info\_t, 196  
dual\_media\_fiber\_speed  
    port\_custom\_conf\_t, 28  
duneq\_p  
    vtss\_ewis\_defects\_s, 90  
dwrr\_cnt  
    vtss\_qos\_port\_conf\_t, 458  
dwrr\_enable  
    vtss\_qos\_port\_conf\_t, 457  
ebs  
    vtss\_dlb\_policer\_conf\_t, 74  
    vtss\_shaper\_t, 471  
edc\_fw\_api\_load  
    vtss\_phy\_10g\_fw\_status\_t, 231  
edc\_fw\_chksum  
    vtss\_phy\_10g\_fw\_status\_t, 230  
edc\_fw\_load  
    vtss\_phy\_10g\_mode\_t, 255  
edc\_fw\_rev

vtss\_phy\_10g\_fw\_status\_t, 230  
 eee\_ena  
   vtss\_eee\_port\_conf\_t, 75  
 eee\_ena\_phy  
   vtss\_phy\_eee\_conf\_t, 310  
 eee\_fast\_queues  
   vtss\_eee\_port\_conf\_t, 75  
 eee\_mode  
   vtss\_phy\_eee\_conf\_t, 309  
 egr\_fcs\_err  
   vtss\_phy\_ts\_stats\_t, 383  
 egr\_frm\_mod\_cnt  
   vtss\_phy\_ts\_stats\_t, 384  
 egr\_pream\_shrink\_err  
   vtss\_phy\_ts\_stats\_t, 383  
 eir  
   vtss\_dlb\_policer\_conf\_t, 74  
   vtss\_shaper\_t, 471  
 en\_ob  
   vtss\_phy\_10g\_base\_kr\_conf\_t, 213  
 ena\_ifh\_header  
   vtss\_port\_ifh\_t, 416  
 ena\_inj\_header  
   vtss\_port\_ifh\_t, 416  
 ena\_xtr\_header  
   vtss\_port\_ifh\_t, 416  
 enable  
   port\_custom\_conf\_t, 27  
   vtss\_ace\_ptp\_t, 50  
   vtss\_ace\_sip\_smac\_t, 51  
   vtss\_dlb\_policer\_conf\_t, 73  
   vtss\_fdma\_cfg\_t, 122  
   vtss\_mac\_table\_entry\_t, 158  
   vtss\_npi\_conf\_t, 163  
   vtss\_packet\_rx\_queue\_npi\_conf\_t, 192  
   vtss\_phy\_10g\_auto\_failover\_conf\_t, 210  
   vtss\_phy\_10g\_base\_kr\_training\_t, 218  
   vtss\_phy\_10g\_ckout\_conf\_t, 221  
   vtss\_phy\_10g\_clause\_37\_control\_t, 225  
   vtss\_phy\_10g\_kr\_status\_fec\_t, 246  
   vtss\_phy\_10g\_lane\_sync\_conf\_t, 248  
   vtss\_phy\_10g\_loopback\_t, 251  
   vtss\_phy\_10g\_pkt\_gen\_conf\_t, 268  
   vtss\_phy\_10g\_pkt\_mon\_conf\_t, 272  
   vtss\_phy\_10g\_prbs\_gen\_conf\_t, 275  
   vtss\_phy\_10g\_prbs\_mon\_conf\_t, 277  
   vtss\_phy\_10g\_sckout\_conf\_t, 283  
   vtss\_phy\_10g\_srefclk\_mode\_t, 291  
   vtss\_phy\_10g\_vscope\_conf\_t, 296  
   vtss\_phy\_ltc\_freq\_synth\_s, 316  
   vtss\_phy\_ts\_generic\_action\_t, 354  
   vtss\_phy\_ts\_nphase\_status\_t, 370  
   vtss\_phy\_ts\_oam\_engine\_action\_t, 372  
   vtss\_phy\_ts\_ptp\_engine\_action\_t, 378  
   vtss\_port\_clause\_37\_control\_t, 399  
   vtss\_red\_v2\_t, 460  
   vtss\_sync\_clock\_in\_t, 474  
   vtss\_sync\_clock\_out\_t, 475  
     vtss\_ts\_ext\_clock\_mode\_t, 481  
 enable\_pass\_thru  
   vtss\_phy\_10g\_clause\_37\_control\_t, 225  
   vtss\_phy\_10g\_mode\_t, 263  
 enabled  
   vtss\_sgpi\_port\_conf\_t, 469  
 encaps\_type  
   vtss\_phy\_ts\_eng\_init\_conf\_t, 337  
 end  
   vtss\_phy\_ts\_mpls\_conf\_t, 368  
 eng\_minE  
   vtss\_phy\_ts\_fifo\_conf\_t, 349  
 eng\_mode  
   vtss\_phy\_ts\_engine\_flow\_conf\_t, 340  
 eng\_recov  
   vtss\_phy\_ts\_fifo\_conf\_t, 348  
 eng\_used  
   vtss\_phy\_ts\_eng\_init\_conf\_t, 337  
 err\_blk\_cnt  
   vtss\_phy\_pcs\_cnt\_t, 323  
 error\_counter  
   vtss\_phy\_10g\_pcs\_prbs\_mon\_conf\_t, 267  
 error\_free\_x  
   vtss\_phy\_10g\_vscope\_scan\_status\_t, 300  
 error\_free\_y  
   vtss\_phy\_10g\_vscope\_scan\_status\_t, 300  
 error\_states  
   vtss\_phy\_10g\_prbs\_mon\_conf\_t, 278  
 error\_status  
   vtss\_phy\_10g\_prbs\_mon\_conf\_t, 279  
 error\_thres  
   vtss\_phy\_10g\_vscope\_conf\_t, 296  
 errors  
   vtss\_phy\_10g\_vscope\_scan\_status\_t, 300  
 eth1\_opt  
   vtss\_phy\_ts\_generic\_flow\_conf\_t, 356  
   vtss\_phy\_ts\_oam\_engine\_flow\_conf\_t, 373  
   vtss\_phy\_ts\_ptp\_engine\_flow\_conf\_t, 380  
 eth2\_opt  
   vtss\_phy\_ts\_oam\_engine\_flow\_conf\_t, 374  
   vtss\_phy\_ts\_ptp\_engine\_flow\_conf\_t, 380  
 ethernet  
   vtss\_ace\_frame\_arp\_t, 35  
 ethernet\_like  
   vtss\_port\_counters\_t, 405  
 etype  
   vtss\_ace\_frame\_etype\_t, 37  
   vtss\_ace\_t, 54  
   vtss\_packet\_rx\_meta\_t, 185  
   vtss\_phy\_10g\_pkt\_gen\_conf\_t, 269  
   vtss\_phy\_ts\_eth\_conf\_t, 343  
   vtss\_qce\_frame\_etype\_t, 435  
   vtss\_qce\_key\_t, 442  
   vtss\_vce\_frame\_etype\_t, 500  
   vtss\_vce\_key\_t, 506  
 evnt  
   vtss\_phy\_10g\_auto\_failover\_conf\_t, 208  
 ewis\_cnt\_cfg

vtss\_ewis\_static\_conf\_s, 108  
ewis\_cntr\_thresh\_conf  
    vtss\_ewis\_conf\_s, 84  
ewis\_init\_done  
    vtss\_ewis\_conf\_s, 81  
ewis\_lof\_ctrl1  
    vtss\_ewis\_static\_conf\_s, 107  
ewis\_lof\_ctrl2  
    vtss\_ewis\_static\_conf\_s, 107  
ewis\_mode  
    vtss\_ewis\_conf\_s, 82  
ewis\_mode\_ctrl  
    vtss\_ewis\_static\_conf\_s, 106  
ewis\_pmtick\_ctrl  
    vtss\_ewis\_static\_conf\_s, 108  
ewis\_rx\_ctrl1  
    vtss\_ewis\_static\_conf\_s, 105  
ewis\_rx\_err\_frc1  
    vtss\_ewis\_static\_conf\_s, 107  
ewis\_rx\_frm\_ctrl1  
    vtss\_ewis\_static\_conf\_s, 107  
ewis\_rx\_frm\_ctrl2  
    vtss\_ewis\_static\_conf\_s, 107  
ewis\_tx\_a1\_a2  
    vtss\_ewis\_static\_conf\_s, 106  
ewis\_tx\_c2\_h1  
    vtss\_ewis\_static\_conf\_s, 106  
ewis\_tx\_z0\_e1  
    vtss\_ewis\_static\_conf\_s, 106  
ewis\_txctrl1  
    vtss\_ewis\_static\_conf\_s, 105  
ewis\_txctrl2  
    vtss\_ewis\_static\_conf\_s, 105  
exc\_col\_cont  
    port\_custom\_conf\_t, 28  
    vtss\_port\_conf\_t, 403  
exp\_sl  
    vtss\_ewis\_conf\_s, 83  
exsl  
    vtss\_ewis\_sl\_conf\_s, 104  
external  
    vtss\_irq\_conf\_t, 148  
  
f\_ebc\_thr\_l  
    vtss\_ewis\_counter\_threshold\_s, 88  
f\_ebc\_thr\_p  
    vtss\_ewis\_counter\_threshold\_s, 88  
FALSE  
    types.h, 578  
family  
    vtss\_phy\_10g\_id\_t, 240  
fan\_low\_pol  
    vtss\_fan\_conf\_t, 120  
fan\_open\_col  
    vtss\_fan\_conf\_t, 121  
fan\_pwm\_freq  
    vtss\_fan\_conf\_t, 120  
far\_end\_enable  
    vtss\_phy\_loopback\_t, 314  
fast\_link\_stat\_ena  
    vtss\_phy\_mac\_serdes\_pcs\_ctrl\_t, 319  
fault  
    vtss\_ewis\_cons\_act\_s, 85  
    vtss\_ewis\_status\_s, 109  
fault\_on\_aisl  
    vtss\_ewis\_fault\_cons\_act\_s, 94  
fault\_on\_aisp  
    vtss\_ewis\_fault\_cons\_act\_s, 94  
fault\_on\_feaisp  
    vtss\_ewis\_fault\_cons\_act\_s, 93  
fault\_on\_fplmp  
    vtss\_ewis\_fault\_cons\_act\_s, 93  
fault\_on\_lcdp  
    vtss\_ewis\_fault\_cons\_act\_s, 94  
fault\_on\_lof  
    vtss\_ewis\_fault\_cons\_act\_s, 93  
fault\_on\_lopp  
    vtss\_ewis\_fault\_cons\_act\_s, 94  
fault\_on\_los  
    vtss\_ewis\_fault\_cons\_act\_s, 93  
fault\_on\_plmp  
    vtss\_ewis\_fault\_cons\_act\_s, 94  
fault\_on\_rdil  
    vtss\_ewis\_fault\_cons\_act\_s, 93  
fault\_on\_sef  
    vtss\_ewis\_fault\_cons\_act\_s, 93  
fcs  
    vtss\_packet\_rx\_meta\_t, 185  
fdx  
    port\_custom\_conf\_t, 27  
    vtss\_phy\_10g\_clause\_37\_adv\_t, 222  
    vtss\_phy\_forced\_t, 312  
    vtss\_port\_clause\_37\_adv\_t, 398  
    vtss\_port\_conf\_t, 402  
    vtss\_port\_sgmii\_aneg\_t, 428  
    vtss\_port\_status\_t, 430  
fdx\_gap  
    vtss\_port\_frame\_gaps\_t, 411  
fec  
    vtss\_phy\_10g\_base\_kr\_status\_t, 216  
fec\_abil  
    vtss\_phy\_10g\_base\_kr\_id\_adv\_abil\_t, 214  
fec\_enable  
    vtss\_phy\_10g\_kr\_status\_aneg\_t, 244  
fec\_req  
    vtss\_phy\_10g\_base\_kr\_id\_adv\_abil\_t, 215  
fiber  
    vtss\_port\_status\_t, 431  
file  
    vtss\_api\_lock\_t, 66  
fill\_level  
    vtss\_eee\_port\_counter\_t, 77  
fill\_level\_get  
    vtss\_eee\_port\_counter\_t, 77  
fill\_level\_thres

vtss\_eee\_port\_counter\_t, 77  
 filter  
   vtss\_packet\_port\_filter\_t, 168  
   vtss\_phy\_10g\_auto\_failover\_conf\_t, 210  
 flf  
   vtss\_phy\_conf\_t, 307  
 flow\_conf  
   vtss\_phy\_ts\_engine\_flow\_conf\_t, 341  
 flow\_control  
   port\_custom\_conf\_t, 27  
   vtss\_policer\_ext\_t, 395  
   vtss\_port\_conf\_t, 402  
 flow\_en  
   vtss\_phy\_ts\_eth\_conf\_t, 343  
   vtss\_phy\_ts\_gen\_conf\_t, 352  
   vtss\_phy\_ts\_ip\_conf\_t, 363  
   vtss\_phy\_ts\_mpls\_conf\_t, 366  
 flow\_end\_index  
   vtss\_phy\_ts\_eng\_init\_conf\_t, 338  
 flow\_id  
   vtss\_phy\_ts\_generic\_action\_t, 354  
 flow\_match\_mode  
   vtss\_phy\_ts\_eng\_init\_conf\_t, 337  
 flow\_offset  
   vtss\_phy\_ts\_gen\_conf\_t, 352  
 flow\_opt  
   vtss\_phy\_ts\_eth\_conf\_t, 347  
   vtss\_phy\_ts\_gen\_conf\_t, 353  
   vtss\_phy\_ts\_ip\_conf\_t, 365  
   vtss\_phy\_ts\_mpls\_conf\_t, 369  
 flow\_st\_index  
   vtss\_phy\_ts\_eng\_init\_conf\_t, 337  
 fltr\_val  
   vtss\_phy\_10g\_auto\_failover\_conf\_t, 210  
 force  
   vtss\_phy\_reset\_conf\_t, 326  
 force\_adv\_ability  
   vtss\_phy\_mac\_serdes\_pcs\_ctrl\_t, 318  
   vtss\_phy\_media\_serdes\_pcs\_ctrl\_t, 321  
 force\_ais  
   vtss\_ewis\_line\_force\_mode\_s, 96  
   vtss\_ewis\_line\_tx\_force\_mode\_s, 97  
 force\_ams\_sel  
   vtss\_phy\_conf\_t, 308  
 force\_fefi  
   vtss\_phy\_media\_serdes\_pcs\_ctrl\_t, 322  
 force\_fefi\_value  
   vtss\_phy\_media\_serdes\_pcs\_ctrl\_t, 322  
 force\_hls  
   vtss\_phy\_media\_serdes\_pcs\_ctrl\_t, 322  
 force\_mode  
   vtss\_ewis\_conf\_s, 82  
 force\_rdi  
   vtss\_ewis\_line\_force\_mode\_s, 97  
   vtss\_ewis\_line\_tx\_force\_mode\_s, 98  
   vtss\_ewis\_path\_force\_mode\_s, 99  
 force\_uneq  
   vtss\_ewis\_path\_force\_mode\_s, 98  
 forced  
   vtss\_phy\_conf\_t, 306  
 frag  
   vtss\_phy\_10g\_pkt\_mon\_conf\_t, 273  
 fragment  
   vtss\_ace\_frame\_ipv4\_t, 38  
   vtss\_qce\_frame\_ipv4\_t, 436  
   vtss\_vce\_frame\_ipv4\_t, 501  
 frame  
   vtss\_ace\_t, 55  
   vtss\_qce\_key\_t, 443  
   vtss\_vce\_key\_t, 507  
 frame\_gaps  
   vtss\_port\_conf\_t, 401  
 frame\_length\_chk  
   port\_custom\_conf\_t, 29  
   vtss\_port\_conf\_t, 403  
 frame\_rate  
   vtss\_policer\_ext\_t, 393  
 frame\_single  
   vtss\_phy\_10g\_pkt\_gen\_conf\_t, 269  
 frame\_type  
   vtss\_vlan\_port\_conf\_t, 515  
 frames  
   vtss\_phy\_10g\_pkt\_gen\_conf\_t, 269  
 freeze  
   vtss\_phy\_10g\_apc\_status\_t, 207  
 freeze\_bit\_mask  
   vtss\_phy\_10g\_ib\_conf\_t, 236  
 freq  
   vtss\_phy\_10g\_ckout\_conf\_t, 220  
   vtss\_phy\_10g\_sckout\_conf\_t, 283  
   vtss\_phy\_10g\_srefclk\_mode\_t, 291  
   vtss\_phy\_clock\_conf\_t, 304  
   vtss\_ts\_ext\_clock\_mode\_t, 481  
 frm\_len  
   vtss\_packet\_tx\_info\_t, 197  
 frm\_limit\_per\_tick  
   vtss\_fdma\_throttle\_cfg\_t, 131  
 frm\_ptr  
   tag\_vtss\_fdma\_list, 31  
 frst\_lvl\_after\_top  
   vtss\_phy\_ts\_mpls\_conf\_t, 367  
 frst\_lvl\_before\_end  
   vtss\_phy\_ts\_mpls\_conf\_t, 368  
 fsm\_lock  
   vtss\_lcpll\_status\_t, 153  
 fsm\_stat  
   vtss\_lcpll\_status\_t, 153  
 full  
   vtss\_debug\_info\_t, 70  
 function  
   vtss\_api\_lock\_t, 66  
 fwd\_enable  
   vtss\_mirror\_conf\_t, 161  
 fwd\_mode  
   vtss\_vstax\_tx\_header\_t, 528  
 gain

vtss\_phy\_10g\_ib\_conf\_t, 233  
gain\_stat  
    vtss\_lcpoll\_status\_t, 153  
gainadj  
    vtss\_phy\_10g\_ib\_conf\_t, 233  
garp\_cpu\_only  
    vtss\_packet\_rx\_reg\_t, 193  
garp\_queue  
    vtss\_packet\_rx\_queue\_map\_t, 190  
garp\_reg  
    vtss\_packet\_rx\_port\_conf\_t, 188  
gen  
    vtss\_phy\_ts\_engine\_flow\_conf\_t, 341  
gen\_conf  
    vtss\_phy\_ts\_engine\_action\_t, 339  
gen\_opt  
    vtss\_phy\_ts\_generic\_flow\_conf\_t, 356  
generate  
    vtss\_port\_flow\_control\_conf\_t, 410  
generate\_pause  
    vtss\_aneg\_t, 65  
glag\_no  
    vtss\_packet\_frame\_info\_t, 166  
    vtss\_packet\_port\_info\_t, 169  
    vtss\_packet\_rx\_info\_t, 175  
    vtss\_vstax\_rx\_header\_t, 526  
    vtss\_vstax\_tx\_header\_t, 529  
good\_crc  
    vtss\_phy\_10g\_pkt\_mon\_conf\_t, 272  
group  
    vtss\_debug\_info\_t, 69  
group\_id  
    vtss\_vlan\_trans\_grp2vlan\_conf\_t, 517  
    vtss\_vlan\_trans\_port2grp\_conf\_t, 518  
grp\_map  
    vtss\_packet\_rx\_conf\_t, 171  
  
h\_apc\_conf  
    vtss\_phy\_10g\_mode\_t, 262  
h\_clk\_src  
    vtss\_phy\_10g\_mode\_t, 261  
h\_ib\_conf  
    vtss\_phy\_10g\_mode\_t, 262  
h\_media  
    vtss\_phy\_10g\_mode\_t, 262  
h\_offset\_guard  
    vtss\_phy\_10g\_mode\_t, 259  
h\_pcs  
    vtss\_phy\_10g\_serdes\_status\_t, 290  
h\_pll5g\_fsm\_lock  
    vtss\_phy\_10g\_serdes\_status\_t, 285  
h\_pll5g\_fsm\_stat  
    vtss\_phy\_10g\_serdes\_status\_t, 285  
h\_pll5g\_gain  
    vtss\_phy\_10g\_serdes\_status\_t, 285  
h\_pll5g\_lock\_status  
    vtss\_phy\_10g\_serdes\_status\_t, 285  
h\_pma  
    vtss\_phy\_10g\_serdes\_status\_t, 289  
  
h\_rx\_rcpll\_fsm\_status  
    vtss\_phy\_10g\_serdes\_status\_t, 287  
h\_rx\_rcpll\_lock\_status  
    vtss\_phy\_10g\_serdes\_status\_t, 286  
h\_rx\_rcpll\_range  
    vtss\_phy\_10g\_serdes\_status\_t, 286  
h\_rx\_rcpll\_vco\_load  
    vtss\_phy\_10g\_serdes\_status\_t, 287  
h\_tx\_rcpll\_fsm\_status  
    vtss\_phy\_10g\_serdes\_status\_t, 288  
h\_tx\_rcpll\_lock\_status  
    vtss\_phy\_10g\_serdes\_status\_t, 288  
h\_tx\_rcpll\_range  
    vtss\_phy\_10g\_serdes\_status\_t, 288  
h\_tx\_rcpll\_vco\_load  
    vtss\_phy\_10g\_serdes\_status\_t, 288  
hdx  
    vtss\_phy\_10g\_clause\_37\_adv\_t, 222  
    vtss\_port\_clause\_37\_adv\_t, 398  
    vtss\_port\_sgmii\_aneg\_t, 428  
hdx\_gap\_1  
    vtss\_port\_frame\_gaps\_t, 411  
hdx\_gap\_2  
    vtss\_port\_frame\_gaps\_t, 411  
header  
    vtss\_ace\_ptp\_t, 50  
high  
    vtss\_phy\_timestamp\_t, 332  
    vtss\_vcap\_udp\_tcp\_t, 495  
    vtss\_vcap\_vr\_t, 498  
high\_ber\_latched  
    vtss\_phy\_pcs\_cnt\_t, 323  
high\_duty\_cycle  
    vtss\_phy\_ltc\_freq\_synth\_s, 316  
high\_input\_gain  
    vtss\_phy\_10g\_mode\_t, 254  
hints  
    vtss\_packet\_rx\_info\_t, 174  
hl\_clk\_synth  
    vtss\_phy\_10g\_mode\_t, 254  
host  
    vtss\_phy\_10g\_clause\_37\_cmn\_status\_t, 224  
    vtss\_phy\_10g\_clause\_37\_control\_t, 225  
host\_kr  
    vtss\_phy\_10g\_base\_kr\_train\_aneg\_t, 217  
host\_rx  
    vtss\_phy\_10g\_polarity\_inv\_t, 274  
host\_tx  
    vtss\_phy\_10g\_polarity\_inv\_t, 274  
hpcs  
    vtss\_phy\_10g\_status\_t, 293  
hpcs\_1g  
    vtss\_phy\_10g\_status\_t, 293  
hpma  
    vtss\_phy\_10g\_status\_t, 292  
hw\_cnt  
    vtss\_os\_timestamp\_t, 164  
hw\_tstamp

vtss\_packet\_rx\_info\_t, 180  
 hw\_tstamp\_decoded  
     vtss\_packet\_rx\_info\_t, 180  
  
 i16  
     types.h, 595  
 i32  
     types.h, 595  
 i64  
     types.h, 595  
 i8  
     types.h, 595  
 i\_cpu\_en  
     vtss\_phy\_reset\_conf\_t, 327  
 i\_tag  
     vtss\_phy\_ts\_eth\_conf\_t, 347  
 ib\_conf  
     vtss\_phy\_10g\_ib\_status\_t, 237  
     vtss\_phy\_10g\_mode\_t, 256  
 ib\_cterm\_ena  
     vtss\_serdes\_macro\_conf\_t, 465  
 ib\_ini\_lp  
     vtss\_phy\_10g\_mode\_t, 257  
 ib\_max\_lp  
     vtss\_phy\_10g\_mode\_t, 258  
 ib\_min\_lp  
     vtss\_phy\_10g\_mode\_t, 258  
 ib\_par\_cfg, 25  
     max, 26  
     min, 25  
     value, 25  
 ib\_storage  
     vtss\_phy\_10g\_ib\_storage\_t, 238  
 ib\_storage\_bool  
     vtss\_phy\_10g\_ib\_storage\_t, 238  
 icpu\_activity  
     vtss\_phy\_10g\_fw\_status\_t, 231  
 id  
     vtss\_ace\_t, 52  
     vtss\_qce\_t, 445  
     vtss\_ts\_timestamp\_t, 486  
     vtss\_vce\_t, 509  
 ietf\_oam\_conf  
     vtss\_phy\_ts\_oam\_engine\_action\_t, 372  
 if\_group  
     vtss\_port\_counters\_t, 405  
 if\_type  
     vtss\_port\_conf\_t, 401  
 ifInBroadcastPkts  
     vtss\_port\_if\_group\_counters\_t, 413  
 ifInDiscards  
     vtss\_port\_if\_group\_counters\_t, 413  
 ifInErrors  
     vtss\_port\_if\_group\_counters\_t, 414  
 ifInMulticastPkts  
     vtss\_port\_if\_group\_counters\_t, 413  
 ifInNUcastPkts  
     vtss\_port\_if\_group\_counters\_t, 413  
 ifInOctets  
     vtss\_port\_if\_group\_counters\_t, 412  
 ifInUcastPkts  
     vtss\_port\_if\_group\_counters\_t, 413  
 ifOutBroadcastPkts  
     vtss\_port\_if\_group\_counters\_t, 414  
 ifOutDiscards  
     vtss\_port\_if\_group\_counters\_t, 415  
 ifOutErrors  
     vtss\_port\_if\_group\_counters\_t, 415  
 ifOutMulticastPkts  
     vtss\_port\_if\_group\_counters\_t, 414  
 ifOutNUcastPkts  
     vtss\_port\_if\_group\_counters\_t, 415  
 ifOutOctets  
     vtss\_port\_if\_group\_counters\_t, 414  
 ifOutUcastPkts  
     vtss\_port\_if\_group\_counters\_t, 414  
 ifh  
     vtss\_packet\_tx\_ifh\_t, 194  
 igmp\_cpu\_only  
     vtss\_packet\_rx\_reg\_t, 193  
 igmp\_queue  
     vtss\_packet\_rx\_queue\_map\_t, 190  
 in\_range  
     vtss\_vcap\_udp\_tcp\_t, 495  
 in\_sig  
     vtss\_gpio\_10g\_gpio\_mode\_t, 134  
 incomplete  
     vtss\_phy\_10g\_prbs\_mon\_conf\_t, 280  
 incr\_levn  
     vtss\_phy\_10g\_jitter\_conf\_t, 242  
 ingr\_fcs\_err  
     vtss\_phy\_ts\_stats\_t, 383  
 ingr\_frm\_mod\_cnt  
     vtss\_phy\_ts\_stats\_t, 384  
 ingr\_pream\_shrink\_err  
     vtss\_phy\_ts\_stats\_t, 383  
 ingress  
     vtss\_phy\_10g\_pkt\_gen\_conf\_t, 269  
 ingress\_filter  
     vtss\_vlan\_port\_conf\_t, 515  
 inhibit\_odd\_start  
     vtss\_phy\_mac\_serd\_pcs\_cntl\_t, 320  
     vtss\_phy\_media\_serd\_pcs\_cntl\_t, 321  
 inj\_grp\_mask  
     vtss\_fdma\_ch\_cfg\_t, 127  
 inner\_tag  
     vtss\_phy\_ts\_eth\_conf\_t, 347  
     vtss\_qce\_key\_t, 442  
 inner\_tag\_type  
     vtss\_phy\_ts\_eth\_conf\_t, 345  
 input  
     vtss\_gpio\_10g\_gpio\_mode\_t, 133  
 inst  
     vtss\_api\_lock\_t, 66  
 instable  
     vtss\_phy\_10g\_prbs\_mon\_conf\_t, 280  
 int\_fmt

vtss\_ts\_internal\_mode\_t, 483  
int\_pol\_high  
    vtss\_sgpio\_port\_conf\_t, 469  
interface  
    vtss\_phy\_10g\_mode\_t, 253  
ip  
    vtss\_ace\_frame\_arp\_t, 35  
ip1\_opt  
    vtss\_phy\_ts\_ptp\_engine\_flow\_conf\_t, 380  
ip2\_opt  
    vtss\_phy\_ts\_ptp\_engine\_flow\_conf\_t, 380  
ip\_addr  
    vtss\_phy\_ts\_ip\_conf\_t, 365  
ip\_enable  
    vtss\_phy\_ts\_sertod\_conf\_t, 382  
ip\_mode  
    vtss\_phy\_ts\_ip\_conf\_t, 362  
ipg\_len  
    vtss\_phy\_10g\_pkt\_gen\_conf\_t, 270  
ipmc\_ctrl\_cpu\_copy  
    vtss\_packet\_rx\_reg\_t, 193  
ipmc\_ctrl\_queue  
    vtss\_packet\_rx\_queue\_map\_t, 190  
ipv4  
    vtss\_ace\_t, 54  
    vtss\_ip\_addr\_t, 141  
    vtss\_phy\_ts\_ip\_conf\_t, 364  
    vtss\_qce\_key\_t, 443  
    vtss\_vce\_key\_t, 507  
ipv4\_uc  
    vtss\_routing\_entry\_t, 463  
ipv4uc\_received\_frames  
    vtss\_l3\_counters\_t, 150  
ipv4uc\_received\_octets  
    vtss\_l3\_counters\_t, 150  
ipv4uc\_transmitted\_frames  
    vtss\_l3\_counters\_t, 151  
ipv4uc\_transmitted\_octets  
    vtss\_l3\_counters\_t, 151  
ipv6  
    vtss\_ace\_t, 55  
    vtss\_ip\_addr\_t, 141  
    vtss\_phy\_ts\_ip\_conf\_t, 364  
    vtss\_qce\_key\_t, 443  
    vtss\_vce\_key\_t, 507  
ipv6\_uc  
    vtss\_routing\_entry\_t, 463  
ipv6uc\_received\_frames  
    vtss\_l3\_counters\_t, 151  
ipv6uc\_received\_octets  
    vtss\_l3\_counters\_t, 151  
ipv6uc\_transmitted\_frames  
    vtss\_l3\_counters\_t, 152  
ipv6uc\_transmitted\_octets  
    vtss\_l3\_counters\_t, 151  
is\_high\_amp  
    vtss\_phy\_10g\_clk\_src\_t, 228  
is\_host  
    vtss\_phy\_10g\_ib\_conf\_t, 236  
    vtss\_phy\_10g\_ob\_status\_t, 266  
is\_host\_side  
    vtss\_phy\_10g\_auto\_failover\_conf\_t, 209  
is\_host\_wan  
    vtss\_phy\_10g\_mode\_t, 261  
is\_init  
    vtss\_phy\_10g\_mode\_t, 263  
isdx  
    vtss\_packet\_rx\_info\_t, 182  
    vtss\_packet\_tx\_info\_t, 203  
    vtss\_vstax\_rx\_header\_t, 527  
isdx\_dont\_use  
    vtss\_packet\_tx\_info\_t, 204  
keep\_ttl  
    vtss\_vstax\_tx\_header\_t, 529  
key  
    vtss\_qce\_t, 445  
    vtss\_vce\_t, 509  
known\_broadcast  
    vtss\_policer\_ext\_t, 393  
known\_multicast  
    vtss\_policer\_ext\_t, 393  
known\_unicast  
    vtss\_policer\_ext\_t, 393  
  
l  
    vtss\_phy\_10g\_ib\_conf\_t, 234  
l2\_types.h  
    vtss\_sflow\_type\_t, 531  
l\_apc\_conf  
    vtss\_phy\_10g\_mode\_t, 263  
l\_clk\_src  
    vtss\_phy\_10g\_mode\_t, 261  
l\_h  
    vtss\_phy\_10g\_clause\_37\_control\_t, 225  
l\_ib\_conf  
    vtss\_phy\_10g\_mode\_t, 262  
l\_media  
    vtss\_phy\_10g\_mode\_t, 262  
l\_offset\_guard  
    vtss\_phy\_10g\_mode\_t, 259  
l\_pcs  
    vtss\_phy\_10g\_serdes\_status\_t, 290  
l\_pll5g\_fsm\_lock  
    vtss\_phy\_10g\_serdes\_status\_t, 286  
l\_pll5g\_fsm\_stat  
    vtss\_phy\_10g\_serdes\_status\_t, 286  
l\_pll5g\_gain  
    vtss\_phy\_10g\_serdes\_status\_t, 286  
l\_pll5g\_lock\_status  
    vtss\_phy\_10g\_serdes\_status\_t, 285  
l\_pma  
    vtss\_phy\_10g\_serdes\_status\_t, 290  
l\_rx\_rcpll\_fsm\_status  
    vtss\_phy\_10g\_serdes\_status\_t, 288  
l\_rx\_rcpll\_lock\_status  
    vtss\_phy\_10g\_serdes\_status\_t, 287

l\_rx\_rcpll\_range  
     vtss\_phy\_10g\_serdes\_status\_t, 287  
 l\_rx\_rcpll\_vco\_load  
     vtss\_phy\_10g\_serdes\_status\_t, 287  
 l\_tx\_rcpll\_fsm\_status  
     vtss\_phy\_10g\_serdes\_status\_t, 289  
 l\_tx\_rcpll\_lock\_status  
     vtss\_phy\_10g\_serdes\_status\_t, 289  
 l\_tx\_rcpll\_range  
     vtss\_phy\_10g\_serdes\_status\_t, 289  
 l\_tx\_rcpll\_vco\_load  
     vtss\_phy\_10g\_serdes\_status\_t, 289  
 latch\_timestamp  
     vtss\_packet\_tx\_info\_t, 202  
 layer  
     vtss\_debug\_info\_t, 69  
 lb\_type  
     vtss\_phy\_10g\_loopback\_t, 251  
     vtss\_phy\_api.h, 905  
 ld  
     vtss\_phy\_10g\_ib\_conf\_t, 235  
 ld\_abil  
     vtss\_phy\_10g\_base\_kr\_train\_aneg\_t, 217  
 ld\_pre\_init  
     vtss\_phy\_10g\_base\_kr\_training\_t, 219  
 learn  
     vtss\_acl\_action\_t, 58  
     vtss\_packet\_rx\_header\_t, 172  
 learn\_queue  
     vtss\_packet\_rx\_queue\_map\_t, 190  
 learned  
     vtss\_mac\_table\_status\_t, 159  
 learning  
     vtss\_policer\_ext\_t, 394  
     vtss\_vlan\_vid\_conf\_t, 519  
 length  
     vtss\_ace\_frame\_arp\_t, 35  
     vtss\_packet\_rx\_header\_t, 172  
     vtss\_packet\_rx\_info\_t, 175  
     vtss\_packet\_rx\_meta\_t, 186  
     vtss\_packet\_tx\_ifh\_t, 194  
     vtss\_phy\_veriphy\_result\_t, 389  
 level  
     vtss\_phy\_power\_status\_t, 325  
     vtss\_policer\_t, 396  
     vtss\_shaper\_t, 470  
     vtss\_trace\_conf\_t, 479  
 levn  
     vtss\_phy\_10g\_jitter\_conf\_t, 242  
     vtss\_phy\_10g\_ob\_status\_t, 265  
 lfault  
     vtss\_phy\_10g\_pkt\_mon\_conf\_t, 273  
 limit\_cpu\_traffic  
     vtss\_policer\_ext\_t, 395  
 limit\_noncpu\_traffic  
     vtss\_policer\_ext\_t, 395  
 line  
     vtss\_api\_lock\_t, 66  
     vtss\_phy\_10g\_clause\_37\_cmn\_status\_t, 224  
     vtss\_phy\_10g\_clause\_37\_control\_t, 225  
     vtss\_phy\_10g\_prbs\_gen\_conf\_t, 276  
     vtss\_phy\_10g\_prbs\_mon\_conf\_t, 277  
     vtss\_phy\_10g\_vscope\_conf\_t, 296  
     vtss\_phy\_10g\_vscope\_scan\_conf\_t, 297  
 line\_kr  
     vtss\_phy\_10g\_base\_kr\_train\_aneg\_t, 217  
 line\_rate  
     vtss\_dlb\_policer\_conf\_t, 73  
 line\_rx  
     vtss\_phy\_10g\_polarity\_inv\_t, 274  
 line\_rx\_force  
     vtss\_ewis\_force\_mode\_s, 95  
 line\_tx  
     vtss\_phy\_10g\_polarity\_inv\_t, 274  
 line\_tx\_force  
     vtss\_ewis\_force\_mode\_s, 95  
 link  
     vtss\_phy\_10g\_clause\_37\_status\_t, 226  
     vtss\_phy\_veriphy\_result\_t, 389  
     vtss\_port\_sgmii\_aneg\_t, 428  
     vtss\_port\_status\_t, 430  
 link\_6g\_distance  
     vtss\_phy\_10g\_mode\_t, 261  
 link\_change  
     vtss\_intr\_t, 140  
 link\_down  
     vtss\_port\_status\_t, 430  
     vtss\_sublayer\_status\_t, 472  
 link\_stat  
     vtss\_ewis\_status\_s, 109  
 list  
     vtss\_fdma\_ch\_cfg\_t, 128  
 llabs  
     vtss\_os\_ecos.h, 750  
 llc  
     vtss\_ace\_frame\_llc\_t, 48  
     vtss\_ace\_t, 54  
     vtss\_qce\_key\_t, 442  
     vtss\_vce\_key\_t, 506  
 load  
     vtss\_ts\_alt\_clock\_mode\_t, 480  
 lock\_status  
     vtss\_lcpll\_status\_t, 152  
 locked  
     vtss\_mac\_table\_entry\_t, 157  
 log\_message\_interval  
     vtss\_acl\_ptp\_action\_conf\_t, 62  
 loop  
     vtss\_port\_conf\_t, 404  
 loopback  
     vtss\_ewis\_test\_conf\_s, 110  
 lopc\_stat  
     vtss\_phy\_10g\_status\_t, 294  
 low  
     vtss\_phy\_timestamp\_t, 332  
     vtss\_vcap\_udp\_tcp\_t, 495

vtss\_vcap\_vr\_t, 498  
low\_duty\_cycle  
  vtss\_phy\_ltc\_freq\_synth\_s, 316  
lower  
  vtss\_phy\_ts\_eth\_conf\_t, 345  
  vtss\_phy\_ts\_mpls\_lvl\_rng\_t, 369  
  vtss\_phy\_ts\_ptp\_conf\_t, 377  
  vtss\_phy\_ts\_y1731\_oam\_conf\_t, 386  
lp\_advertisement  
  vtss\_eee\_port\_conf\_t, 76  
lp\_anegable  
  vtss\_phy\_10g\_kr\_status\_aneg\_t, 245  
lpc8\_1g  
  vtss\_phy\_10g\_status\_t, 293  
lref\_for\_host  
  vtss\_phy\_10g\_mode\_t, 261  
lrn\_all\_queue  
  vtss\_packet\_rx\_queue\_map\_t, 191  
ls\_inv  
  vtss\_phy\_ts\_sertod\_conf\_t, 382  
ls\_ipbk  
  vtss\_phy\_ts\_alt\_clock\_mode\_s, 336  
ls\_pps\_ipbk  
  vtss\_phy\_ts\_alt\_clock\_mode\_s, 336  
  
MAC\_ADDR\_BROADCAST  
  types.h, 586  
MAX\_CFG\_BUF\_SIZE  
  vtss\_phy\_api.h, 886  
MAX\_STAT\_BUF\_SIZE  
  vtss\_phy\_api.h, 886  
MAX\_WOL\_MAC\_ADDR\_SIZE  
  vtss\_phy\_api.h, 897  
MAX\_WOL\_PASSWD\_SIZE  
  vtss\_phy\_api.h, 897  
MIN\_WOL\_PASSWD\_SIZE  
  vtss\_phy\_api.h, 897  
mac  
  vtss\_qce\_key\_t, 441  
  vtss\_vce\_key\_t, 506  
  vtss\_vid\_mac\_t, 513  
mac\_addr  
  vtss\_phy\_ts\_eth\_conf\_t, 344  
mac\_if  
  vtss\_phy\_reset\_conf\_t, 326  
mac\_if\_pcs  
  vtss\_phy\_conf\_t, 307  
mac\_serdes\_equipment\_enable  
  vtss\_phy\_loopback\_t, 315  
mac\_serdes\_facility\_enable  
  vtss\_phy\_loopback\_t, 314  
mac\_serdes\_input\_enable  
  vtss\_phy\_loopback\_t, 314  
mac\_vid\_queue  
  vtss\_packet\_rx\_queue\_map\_t, 190  
macsec\_ena  
  vtss\_phy\_ts\_init\_conf\_t, 360  
magic\_pkt\_cnt  
  vtss\_phy\_wol\_conf\_t, 391  
  
main\_status  
  vtss\_phy\_10g\_prbs\_mon\_conf\_t, 279  
map  
  vtss\_packet\_rx\_conf\_t, 171  
mask  
  vtss\_phy\_ts\_ach\_conf\_t, 334  
  vtss\_phy\_ts\_eth\_conf\_t, 346, 347  
  vtss\_phy\_ts\_gen\_conf\_t, 353  
  vtss\_phy\_ts\_generic\_action\_t, 355  
  vtss\_phy\_ts\_ip\_conf\_t, 364  
  vtss\_phy\_ts\_ptp\_conf\_t, 376  
  vtss\_phy\_ts\_y1731\_oam\_conf\_t, 386  
  vtss\_vcap\_ip\_t, 487  
  vtss\_vcap\_u128\_t, 488  
  vtss\_vcap\_u16\_t, 489  
  vtss\_vcap\_u24\_t, 490  
  vtss\_vcap\_u32\_t, 491  
  vtss\_vcap\_u40\_t, 492  
  vtss\_vcap\_u48\_t, 493  
  vtss\_vcap\_u8\_t, 494  
  vtss\_vcap\_vid\_t, 496  
  vtss\_vcap\_vr\_t, 497  
masquerade\_port  
  vtss\_packet\_tx\_info\_t, 205  
master  
  vtss\_phy\_10g\_mode\_t, 260  
  vtss\_phy\_conf\_1g\_t, 305  
  vtss\_phy\_status\_1g\_t, 331  
master\_cfg\_fault  
  vtss\_phy\_status\_1g\_t, 331  
match\_mode  
  vtss\_phy\_ts\_ip\_conf\_t, 364  
max  
  ib\_par\_cfg, 26  
  vtss\_red\_v2\_t, 460  
max\_bist\_frames  
  vtss\_phy\_10g\_prbs\_mon\_conf\_t, 277  
max\_bw  
  vtss\_port\_map\_t, 417  
max\_frame\_length  
  vtss\_port\_conf\_t, 402  
max\_length  
  port\_custom\_conf\_t, 28  
max\_tags  
  port\_custom\_conf\_t, 29  
  vtss\_port\_conf\_t, 403  
max\_unit  
  vtss\_red\_v2\_t, 461  
mdi  
  vtss\_phy\_conf\_t, 306  
mdi\_cross  
  vtss\_port\_status\_t, 431  
media\_if  
  vtss\_phy\_reset\_conf\_t, 326  
media\_if\_pcs  
  vtss\_phy\_conf\_t, 307  
media\_mac\_serdes\_crc  
  vtss\_phy\_statistic\_t, 330

media\_mac\_serd़es\_good  
     vtss\_phy\_statistic\_t, 330

media\_serd़es\_equipment\_enable  
     vtss\_phy\_loopback\_t, 315

media\_serd़es\_facility\_enable  
     vtss\_phy\_loopback\_t, 315

media\_serd़es\_input\_enable  
     vtss\_phy\_loopback\_t, 315

meg\_level  
     vtss\_phy\_ts\_y1731\_oam\_conf\_t, 386

miim\_addr  
     vtss\_port\_map\_t, 418

miim\_chip\_no  
     vtss\_port\_map\_t, 418

miim\_controller  
     vtss\_port\_map\_t, 418

miim\_read  
     vtss\_init\_conf\_t, 136

miim\_write  
     vtss\_init\_conf\_t, 136

min  
     ib\_par\_cfg, 25

min\_fl  
     vtss\_red\_v2\_t, 460

mirror  
     vtss\_acl\_action\_t, 59

    vtss\_vlan\_vid\_conf\_t, 520

    vtss\_vstax\_port\_conf\_t, 523

mld\_cpu\_only  
     vtss\_packet\_rx\_reg\_t, 193

mmd\_read  
     vtss\_init\_conf\_t, 136

mmd\_read\_inc  
     vtss\_init\_conf\_t, 137

mmd\_write  
     vtss\_init\_conf\_t, 137

mode  
     vtss\_ewis\_tti\_s, 112

    vtss\_gpio\_10g\_gpio\_mode\_t, 133

    vtss\_phy\_10g\_ckout\_conf\_t, 220

    vtss\_phy\_10g\_host\_clk\_conf\_t, 232

    vtss\_phy\_10g\_line\_clk\_conf\_t, 250

    vtss\_phy\_10g\_rxckout\_conf\_t, 281

    vtss\_phy\_10g\_sckout\_conf\_t, 282

    vtss\_phy\_10g\_txckout\_conf\_t, 295

    vtss\_phy\_conf\_t, 306

    vtss\_phy\_led\_mode\_select\_t, 313

    vtss\_phy\_power\_conf\_t, 324

    vtss\_sgpi\_0\_port\_conf\_t, 469

    vtss\_ts\_operation\_mode\_t, 483

moved  
     vtss\_mac\_table\_status\_t, 160

mpls\_opt  
     vtss\_phy\_ts\_oam\_engine\_flow\_conf\_t, 374

    vtss\_phy\_ts\_ptp\_engine\_flow\_conf\_t, 381

msg\_type  
     vtss\_phy\_ts\_fifo\_sig\_t, 350

mux\_mode

    vtss\_init\_conf\_t, 138

n\_ebc\_thr\_l  
     vtss\_ewis\_counter\_threshold\_s, 87

n\_ebc\_thr\_p  
     vtss\_ewis\_counter\_threshold\_s, 88

n\_ebc\_thr\_s  
     vtss\_ewis\_counter\_threshold\_s, 87

nanoseconds  
     vtss\_phy\_timestamp\_t, 333

    vtss\_timestamp\_t, 478

near\_end\_enable  
     vtss\_phy\_loopback\_t, 314

network  
     vtss\_ipv4\_uc\_t, 144

    vtss\_ipv6\_uc\_t, 147

next  
     tag\_vtss\_fdma\_list, 33

next\_page  
     vtss\_phy\_10g\_clause\_37\_adv\_t, 223

    vtss\_port\_clause\_37\_adv\_t, 399

next\_prot\_offset  
     vtss\_phy\_ts\_gen\_conf\_t, 352

no\_of\_errors  
     vtss\_phy\_10g\_prbs\_mon\_conf\_t, 278

no\_sync  
     vtss\_phy\_10g\_prbs\_mon\_conf\_t, 280

no\_wait  
     vtss\_packet\_rx\_meta\_t, 184

now  
     vtss\_mtimer\_t, 163

npi  
     vtss\_packet\_rx\_queue\_conf\_t, 189

num\_tag  
     vtss\_phy\_ts\_eth\_conf\_t, 344

number  
     vtss\_phy\_led\_mode\_select\_t, 313

oam  
     vtss\_phy\_ts\_engine\_flow\_conf\_t, 341

oam\_conf  
     vtss\_phy\_ts\_engine\_action\_t, 339

    vtss\_phy\_ts\_oam\_engine\_action\_t, 373

oam\_info  
     vtss\_packet\_rx\_info\_t, 181

oam\_info\_decoded  
     vtss\_packet\_rx\_info\_t, 182

oam\_type  
     vtss\_packet\_tx\_info\_t, 203

ob\_conf  
     vtss\_phy\_10g\_mode\_t, 256

ob\_posto  
     serdes\_fields\_t, 30

ob\_sr  
     serdes\_fields\_t, 30

obey  
     vtss\_port\_flow\_control\_conf\_t, 410

obey\_pause  
     vtss\_aneg\_t, 65

offs  
    vtss\_phy\_10g\_ib\_conf\_t, 233  
one\_pps\_in  
    vtss\_ts\_alt\_clock\_mode\_t, 480  
one\_pps\_mode  
    vtss\_ts\_ext\_clock\_mode\_t, 481  
one\_pps\_out  
    vtss\_ts\_alt\_clock\_mode\_t, 479  
one\_step\_txfifo  
    vtss\_phy\_ts\_init\_conf\_t, 361  
op\_enable  
    vtss\_phy\_ts\_sertod\_conf\_t, 382  
op\_mode  
    vtss\_phy\_10g\_apc\_conf\_t, 206  
op\_mode\_flag  
    vtss\_phy\_10g\_apc\_conf\_t, 206  
oper\_mode  
    vtss\_phy\_10g\_mode\_t, 253  
oper\_mode\_t  
    vtss\_phy\_10g\_api.h, 812  
oper\_up  
    port\_custom\_conf\_t, 29  
optimized\_for\_power  
    vtss\_eee\_port\_conf\_t, 76  
options  
    vtss\_ace\_frame\_ipv4\_t, 39  
    vtss\_vce\_frame\_ipv4\_t, 501  
options.h  
    VIPER\_B\_FIFO\_RESET, 551  
    VTSS\_ARCH\_JAGUAR\_2, 534  
    VTSS\_ARCH\_JAGUAR\_2\_B, 534  
    VTSS\_ARCH\_MALIBU\_B, 553  
    VTSS\_ARCH\_MALIBU, 553  
    VTSS\_ARCH\_VENICE\_C, 553  
    VTSS\_CHIP\_10G\_PHY, 534  
    VTSS\_CHIP CU PHY, 551  
    VTSS\_FEATURE\_10GBASE\_KR, 549  
    VTSS\_FEATURE\_10G, 550  
    VTSS\_FEATURE\_ACL\_V2, 546  
    VTSS\_FEATURE\_ACL, 546  
    VTSS\_FEATURE\_AGGR\_GLAG, 537  
    VTSS\_FEATURE\_CLAUSE\_37, 535  
    VTSS\_FEATURE\_EDC\_FW\_LOAD, 552  
    VTSS\_FEATURE\_EEE, 545  
    VTSS\_FEATURE\_EXC\_COL\_CONT, 535  
    VTSS\_FEATURE\_FAN, 545  
    VTSS\_FEATURE\_FDMA, 550  
    VTSS\_FEATURE\_INTERRUPTS, 548  
    VTSS\_FEATURE\_IRQ\_CONTROL, 549  
    VTSS\_FEATURE\_LAYER2, 544  
    VTSS\_FEATURE\_LED\_POW\_REDUC, 548  
    VTSS\_FEATURE\_MAC\_AGE\_AUTO, 545  
    VTSS\_FEATURE\_MAC\_CPU\_QUEUE, 545  
    VTSS\_FEATURE\_MISC, 534  
    VTSS\_FEATURE\_NPI, 548  
    VTSS\_FEATURE\_PACKET\_GROUPING, 544  
    VTSS\_FEATURE\_PACKET\_PORT\_REG, 544  
    VTSS\_FEATURE\_PACKET\_RX, 543  
    VTSS\_FEATURE\_PACKET\_TX, 543  
    VTSS\_FEATURE\_PACKET, 543  
    VTSS\_FEATURE\_PFC, 537  
    VTSS\_FEATURE\_PHY\_TIMESTAMP, 547  
    VTSS\_FEATURE\_PORT\_CNT\_BRIDGE, 536  
    VTSS\_FEATURE\_PORT\_CNT\_ETHER\_LIKE, 536  
    VTSS\_FEATURE\_PORT\_CONTROL, 535  
    VTSS\_FEATURE\_PORT\_IFH, 535  
    VTSS\_FEATURE\_PORT\_MUX, 537  
    VTSS\_FEATURE\_PVLAN, 544  
    VTSS\_FEATURE\_QCL\_KEY\_DIP, 538  
    VTSS\_FEATURE\_QCL\_KEY\_DMAC, 538  
    VTSS\_FEATURE\_QCL\_KEY\_INNER\_TAG, 538  
    VTSS\_FEATURE\_QCL\_KEY\_S\_TAG, 538  
    VTSS\_FEATURE\_QCL\_PCP\_DEI\_ACTION, 538  
    VTSS\_FEATURE\_QCL\_POLICY\_ACTION, 539  
    VTSS\_FEATURE\_QCL\_V2, 537  
    VTSS\_FEATURE\_QCL, 537  
    VTSS\_FEATURE\_QOS\_CLASSIFICATION\_V2, 541  
    VTSS\_FEATURE\_QOS\_DSCP\_CLASS\_DP\_AWARE, 542  
    VTSS\_FEATURE\_QOS\_DSCP\_REMARK\_V2, 542  
    VTSS\_FEATURE\_QOS\_DSCP\_REMARK, 542  
    VTSS\_FEATURE\_QOS\_EGRESS\_QUEUE\_SHAPERS\_APERS, 542  
    VTSS\_FEATURE\_QOS\_EGRESS\_SHAPERS\_DB, 542  
    VTSS\_FEATURE\_QOS\_POLICER\_BC\_SWITCH, 539  
    VTSS\_FEATURE\_QOS\_POLICER\_DLB, 543  
    VTSS\_FEATURE\_QOS\_POLICER\_MC\_SWITCH, 539  
    VTSS\_FEATURE\_QOS\_POLICER\_UC\_SWITCH, 539  
    VTSS\_FEATURE\_QOS\_PORT\_POLICER\_EXT\_DPBL, 540  
    VTSS\_FEATURE\_QOS\_PORT\_POLICER\_EXT\_FPS, 540  
    VTSS\_FEATURE\_QOS\_PORT\_POLICER\_EXT\_FC, 540  
    VTSS\_FEATURE\_QOS\_PORT\_POLICER\_EXT\_TTM\_V2, 540  
    VTSS\_FEATURE\_QOS\_PORT\_POLICER\_EXT, 539  
    VTSS\_FEATURE\_QOS\_QUEUE\_POLICER, 540  
    VTSS\_FEATURE\_QOS\_QUEUE\_TX, 541  
    VTSS\_FEATURE\_QOS\_SCHEDULER\_DWRR\_CNT, 541  
    VTSS\_FEATURE\_QOS\_SCHEDULER\_V2, 541  
    VTSS\_FEATURE\_QOS\_TAG\_REMARK\_V2, 541  
    VTSS\_FEATURE\_QOS\_WRED\_V3, 543  
    VTSS\_FEATURE\_QOS, 536  
    VTSS\_FEATURE\_SERDES\_MACRO\_SETTINGS, 549  
    VTSS\_FEATURE\_SERIAL\_GPIO, 535

VTSS FEATURE\_SFLOW, 549  
 VTSS FEATURE\_SYNCE\_10G, 552  
 VTSS FEATURE\_SYNCE, 548  
 VTSS FEATURE\_TIMESTAMP\_ASYMMETRY\_COMP, 548  
 VTSS FEATURE\_TIMESTAMP\_LATENCY\_CO\_MP, 547  
 VTSS FEATURE\_TIMESTAMP\_ONE\_STEP, 547  
 VTSS FEATURE\_TIMESTAMP\_ORG\_TIME, 547  
 VTSS FEATURE\_TIMESTAMP\_P2P\_DELAY\_COMP, 547  
 VTSS FEATURE\_TIMESTAMP, 546  
 VTSS FEATURE\_VCAP, 546  
 VTSS FEATURE\_VCL, 546  
 VTSS FEATURE\_VLAN\_PORT\_V2, 544  
 VTSS FEATURE\_VLAN\_TRANSLATION, 549  
 VTSS FEATURE\_VLAN\_TX\_TAG, 545  
 VTSS FEATURE\_VSTAX\_V2, 536  
 VTSS FEATURE\_VSTAX, 536  
 VTSS FEATURE\_WARM\_START, 552, 553  
 VTSS FEATURE\_WIS, 553  
 VTSS\_OPT\_FDMA\_DEBUG, 551  
 VTSS\_OPT\_FDMA\_IRQ\_CONTEXT, 551  
 VTSS\_OPT\_PCIE\_ACCESS, 550  
 VTSS\_OPT\_TRACE, 551  
 VTSS\_OPT\_VAUI\_EQ\_CTRL, 552  
 VTSS\_OPT\_VCORE\_III, 550  
 VTSS\_PHY\_10G\_FIFO\_SYNC, 550  
 VTSS\_PHY\_OPT\_VERIPHYS, 552  
 out\_of\_range  
     vtss\_rcpll\_status\_t, 459  
 outer\_tag  
     vtss\_phy\_ts\_eth\_conf\_t, 346  
 outer\_tag\_type  
     vtss\_phy\_ts\_eth\_conf\_t, 345  
 p\_gpio  
     vtss\_gpio\_10g\_gpio\_mode\_t, 134  
 p\_gpio\_intrpt  
     vtss\_gpio\_10g\_gpio\_mode\_t, 135  
 PHASE\_POINTS  
     vtss\_phy\_10g\_api.h, 798  
 PORT\_CAP\_100M\_FDX  
     port.h, 559  
 PORT\_CAP\_100M\_HDX  
     port.h, 559  
 PORT\_CAP\_10G\_FDX  
     port.h, 560  
 PORT\_CAP\_10M\_FDX  
     port.h, 559  
 PORT\_CAP\_10M\_HDX  
     port.h, 558  
 PORT\_CAP\_1G\_FDX  
     port.h, 559  
 PORT\_CAP\_1G\_PHY  
     port.h, 564  
 PORT\_CAP\_2\_5G\_FDX  
     port.h, 559  
 PORT\_CAP\_2\_5G\_TRI\_SPEED\_COPPER  
     port.h, 567  
 PORT\_CAP\_2\_5G\_TRI\_SPEED\_FDX  
     port.h, 567  
 PORT\_CAP\_2\_5G\_TRI\_SPEED  
     port.h, 567  
 PORT\_CAP\_5G\_FDX  
     port.h, 560  
 PORT\_CAP\_ANY\_FIBER  
     port.h, 565  
 PORT\_CAP\_AUTONEG  
     port.h, 558  
 PORT\_CAP\_COPPER  
     port.h, 560  
 PORT\_CAP\_DUAL\_COPPER\_100FX  
     port.h, 563  
 PORT\_CAP\_DUAL\_COPPER  
     port.h, 561  
 PORT\_CAP\_DUAL\_FIBER\_1000X  
     port.h, 566  
 PORT\_CAP\_DUAL\_FIBER\_100FX  
     port.h, 562  
 PORT\_CAP\_DUAL\_FIBER  
     port.h, 561  
 PORT\_CAP\_DUAL\_SFP\_DETECT  
     port.h, 563  
 PORT\_CAP\_FIBER  
     port.h, 560  
 PORT\_CAP\_FLOW\_CTRL  
     port.h, 560  
 PORT\_CAP\_HDX  
     port.h, 563  
 PORT\_CAP\_NONE  
     port.h, 558  
 PORT\_CAP\_SD\_ENABLE  
     port.h, 561  
 PORT\_CAP\_SD\_HIGH  
     port.h, 561  
 PORT\_CAP\_SD\_INTERNAL  
     port.h, 561  
 PORT\_CAP\_SFP\_1G  
     port.h, 566  
 PORT\_CAP\_SFP\_2\_5G  
     port.h, 566  
 PORT\_CAP\_SFP\_DETECT  
     port.h, 562  
 PORT\_CAP\_SFP\_ONLY  
     port.h, 563  
 PORT\_CAP\_SFP\_SD\_HIGH  
     port.h, 567  
 PORT\_CAP\_SPEED\_DUAL\_ANY\_FIBER\_FIXED\_SPEED  
     port.h, 565  
 PORT\_CAP\_SPEED\_DUAL\_ANY\_FIBER  
     port.h, 565  
 PORT\_CAP\_STACKING  
     port.h, 562  
 PORT\_CAP\_TRI\_SPEED\_COPPER  
     port.h, 564

PORT\_CAP\_TRI\_SPEED\_DUAL\_ANY\_FIBER\_FIXE←  
    D\_SFP\_SPEED  
    port.h, 566  
PORT\_CAP\_TRI\_SPEED\_DUAL\_ANY\_FIBER  
    port.h, 566  
PORT\_CAP\_TRI\_SPEED\_DUAL\_COPPER  
    port.h, 564  
PORT\_CAP\_TRI\_SPEED\_DUAL\_FIBER\_100FX  
    port.h, 565  
PORT\_CAP\_TRI\_SPEED\_DUAL\_FIBER  
    port.h, 565  
PORT\_CAP\_TRI\_SPEED\_FDX  
    port.h, 563  
PORT\_CAP\_TRI\_SPEED\_FIBER  
    port.h, 564  
PORT\_CAP\_TRI\_SPEED  
    port.h, 564  
PORT\_CAP\_VTSS\_10G\_PHY  
    port.h, 562  
PORT\_CAP\_XAUI\_LANE\_FLIP  
    port.h, 562  
PRBS\_status  
    vtss\_phy\_10g\_prbs\_mon\_conf\_t, 279  
PRIl64  
    types.h, 576  
PRIu64  
    types.h, 576  
PRIx64  
    types.h, 576  
part\_number  
    vtss\_chip\_id\_t, 68  
    vtss\_phy\_10g\_id\_t, 239  
    vtss\_phy\_type\_t, 387  
partner\_advertisement  
    vtss\_phy\_10g\_clause\_37\_status\_t, 227  
passwd  
    vtss\_secure\_on\_passwd\_t, 464  
path\_force  
    vtss\_ewis\_force\_mode\_s, 96  
path\_txti  
    vtss\_ewis\_conf\_s, 83  
pbb\_en  
    vtss\_phy\_ts\_eth\_conf\_t, 343  
pcp  
    vtss\_mirror\_conf\_t, 161  
    vtss\_qce\_action\_t, 434  
    vtss\_qce\_tag\_t, 447  
    vtss\_vce\_tag\_t, 511  
    vtss\_vlan\_tag\_t, 516  
pcp\_dei\_enable  
    vtss\_qce\_action\_t, 433  
pcs  
    vtss\_phy\_10g\_cnt\_t, 228  
    vtss\_phy\_10g\_status\_t, 292  
pd\_enable  
    vtss\_phy\_mac\_serdes\_pcs\_ctrl\_t, 318  
pdu\_offset  
    vtss\_packet\_tx\_info\_t, 205  
perf\_mode  
    vtss\_ewis\_conf\_s, 84  
pf\_ebc\_l  
    vtss\_ewis\_counter\_s, 86  
    vtss\_ewis\_perf\_s, 101  
pf\_ebc\_mode\_l  
    vtss\_ewis\_perf\_mode\_s, 100  
pf\_ebc\_mode\_p  
    vtss\_ewis\_perf\_mode\_s, 100  
pf\_ebc\_p  
    vtss\_ewis\_counter\_s, 86  
    vtss\_ewis\_perf\_s, 102  
pfc  
    port\_custom\_conf\_t, 27  
    vtss\_port\_flow\_control\_conf\_t, 410  
phy.h  
    VTSS\_PHY\_POWER\_ACTIPHY\_BIT, 554  
    VTSS\_PHY\_POWER\_DYNAMIC\_BIT, 555  
    vtss\_phy\_power\_mode\_t, 555  
    vtss\_phy\_veriphy\_status\_t, 555  
phy\_api\_base\_no  
    vtss\_phy\_10g\_id\_t, 240  
    vtss\_phy\_type\_t, 388  
pi  
    vtss\_init\_conf\_t, 138  
pkt\_len  
    vtss\_phy\_10g\_pkt\_gen\_conf\_t, 270  
pkt\_mode  
    vtss\_phy\_reset\_conf\_t, 327  
pma  
    vtss\_phy\_10g\_status\_t, 292  
pma\_txratecontrol  
    vtss\_phy\_10g\_mode\_t, 259  
pn\_ebc\_l  
    vtss\_ewis\_counter\_s, 86  
    vtss\_ewis\_perf\_s, 101  
pn\_ebc\_mode\_l  
    vtss\_ewis\_perf\_mode\_s, 100  
pn\_ebc\_mode\_p  
    vtss\_ewis\_perf\_mode\_s, 100  
pn\_ebc\_mode\_s  
    vtss\_ewis\_perf\_mode\_s, 99  
pn\_ebc\_p  
    vtss\_ewis\_counter\_s, 86  
    vtss\_ewis\_perf\_s, 102  
pn\_ebc\_s  
    vtss\_ewis\_counter\_s, 86  
    vtss\_ewis\_perf\_s, 101  
polarity  
    vtss\_phy\_10g\_mode\_t, 260  
police  
    vtss\_acl\_action\_t, 58  
policer\_bc  
    vtss\_qos\_conf\_t, 452  
policer\_bc\_frame\_rate  
    vtss\_qos\_conf\_t, 452  
policer\_bc\_mode  
    vtss\_qos\_conf\_t, 452

policer\_ext\_port  
     vtss\_qos\_port\_conf\_t, 453  
 policer\_mc  
     vtss\_qos\_conf\_t, 451  
 policer\_mc\_frame\_rate  
     vtss\_qos\_conf\_t, 451  
 policer\_mc\_mode  
     vtss\_qos\_conf\_t, 451  
 policer\_no  
     vtss\_acl\_action\_t, 58  
 policer\_port  
     vtss\_qos\_port\_conf\_t, 453  
 policer\_queue  
     vtss\_qos\_port\_conf\_t, 454  
 policer\_uc  
     vtss\_qos\_conf\_t, 450  
 policer\_uc\_frame\_rate  
     vtss\_qos\_conf\_t, 451  
 policer\_uc\_mode  
     vtss\_qos\_conf\_t, 451  
 policy  
     vtss\_ace\_t, 52  
 policy\_no  
     vtss\_acl\_port\_conf\_t, 61  
     vtss\_qce\_action\_t, 434  
     vtss\_vce\_action\_t, 499  
 policy\_no\_enable  
     vtss\_qce\_action\_t, 434  
 port  
     vtss\_gpio\_10g\_gpio\_mode\_t, 133  
 port.h  
     PORT\_CAP\_100M\_FDX, 559  
     PORT\_CAP\_100M\_HDX, 559  
     PORT\_CAP\_10G\_FDX, 560  
     PORT\_CAP\_10M\_FDX, 559  
     PORT\_CAP\_10M\_HDX, 558  
     PORT\_CAP\_1G\_FDX, 559  
     PORT\_CAP\_1G\_PHY, 564  
     PORT\_CAP\_2\_5G\_FDX, 559  
     PORT\_CAP\_2\_5G\_TRI\_SPEED\_COPPER, 567  
     PORT\_CAP\_2\_5G\_TRI\_SPEED\_FDX, 567  
     PORT\_CAP\_2\_5G\_TRI\_SPEED, 567  
     PORT\_CAP\_5G\_FDX, 560  
     PORT\_CAP\_ANY\_FIBER, 565  
     PORT\_CAP\_AUTONEG, 558  
     PORT\_CAP\_COPPER, 560  
     PORT\_CAP\_DUAL\_COPPER\_100FX, 563  
     PORT\_CAP\_DUAL\_COPPER, 561  
     PORT\_CAP\_DUAL\_FIBER\_1000X, 566  
     PORT\_CAP\_DUAL\_FIBER\_100FX, 562  
     PORT\_CAP\_DUAL\_FIBER, 561  
     PORT\_CAP\_DUAL\_SFP\_DETECT, 563  
     PORT\_CAP\_FIBER, 560  
     PORT\_CAP\_FLOW\_CTRL, 560  
     PORT\_CAP\_HDX, 563  
     PORT\_CAP\_NONE, 558  
     PORT\_CAP\_SD\_ENABLE, 561  
     PORT\_CAP\_SD\_HIGH, 561  
     PORT\_CAP\_SD\_INTERNAL, 561  
     PORT\_CAP\_SFP\_1G, 566  
     PORT\_CAP\_SFP\_2\_5G, 566  
     PORT\_CAP\_SFP\_DETECT, 562  
     PORT\_CAP\_SFP\_ONLY, 563  
     PORT\_CAP\_SFP\_SD\_HIGH, 567  
     PORT\_CAP\_SPEED\_DUAL\_ANY\_FIBER\_FIXED\_SPEED, 565  
     PORT\_CAP\_SPEED\_DUAL\_ANY\_FIBER, 565  
     PORT\_CAP\_STACKING, 562  
     PORT\_CAP\_TRI\_SPEED\_COPPER, 564  
     PORT\_CAP\_TRI\_SPEED\_DUAL\_ANY\_FIBER\_FIXED\_SFP\_SPEED, 566  
     PORT\_CAP\_TRI\_SPEED\_DUAL\_ANY\_FIBER, 566  
     PORT\_CAP\_TRI\_SPEED\_DUAL\_COPPER, 564  
     PORT\_CAP\_TRI\_SPEED\_DUAL\_FIBER\_100FX, 565  
     PORT\_CAP\_TRI\_SPEED\_DUAL\_FIBER, 565  
     PORT\_CAP\_TRI\_SPEED\_FDX, 563  
     PORT\_CAP\_TRI\_SPEED\_FIBER, 564  
     PORT\_CAP\_TRI\_SPEED, 564  
     PORT\_CAP\_VTSS\_10G\_PHY, 562  
     PORT\_CAP\_XAUI\_LANE\_FLIP, 562  
     port\_cap\_t, 568  
     vtss\_fiber\_port\_speed\_t, 568  
     vtss\_port\_speed\_t, 568  
 port\_0  
     vtss\_vstax\_conf\_t, 521  
 port\_1  
     vtss\_vstax\_conf\_t, 521  
 port\_action  
     vtss\_acl\_action\_t, 58  
 port\_cap\_t  
     port.h, 568  
 port\_cnt  
     vtss\_phy\_type\_t, 388  
 port\_conf  
     vtss\_sgpio\_conf\_t, 468  
 port\_custom\_conf\_t, 26  
     adv\_dis, 28  
     autoneg, 27  
     dual\_media\_fiber\_speed, 28  
     enable, 27  
     exc\_col\_cont, 28  
     fdx, 27  
     flow\_control, 27  
     frame\_length\_chk, 29  
     max\_length, 28  
     max\_tags, 29  
     oper\_up, 29  
     pfc, 27  
     power\_mode, 28  
     speed, 27  
 port\_list  
     vtss\_ace\_t, 52  
     vtss\_acl\_action\_t, 58  
     vtss\_debug\_info\_t, 70

vtss\_qce\_key\_t, 441  
vtss\_vce\_key\_t, 506  
port\_mask  
    vtss\_ts\_timestamp\_alloc\_t, 484  
port\_no  
    vtss\_eps\_port\_conf\_t, 79  
    vtss\_mirror\_conf\_t, 161  
    vtss\_npi\_conf\_t, 164  
    vtss\_packet\_frame\_info\_t, 166  
    vtss\_packet\_port\_info\_t, 169  
    vtss\_packet\_rx\_header\_t, 172  
    vtss\_packet\_rx\_info\_t, 175  
    vtss\_phy\_10g\_auto\_failover\_conf\_t, 208  
    vtss\_sync\_clock\_in\_t, 473  
    vtss\_vstax\_rx\_header\_t, 526  
    vtss\_vstax\_tx\_header\_t, 528  
port\_tx  
    vtss\_packet\_frame\_info\_t, 167  
port\_type  
    vtss\_vlan\_port\_conf\_t, 514  
ports  
    vtss\_vlan\_trans\_port2grp\_conf\_t, 519  
power\_down  
    vtss\_port\_conf\_t, 402  
power\_mode  
    port\_custom\_conf\_t, 28  
ppr  
    vtss\_fan\_conf\_t, 121  
pps\_ls\_lpbk  
    vtss\_phy\_ts\_alt\_clock\_mode\_s, 336  
pps\_offset  
    vtss\_phy\_ts\_pps\_config\_s, 375  
pps\_output\_enable  
    vtss\_phy\_ts\_pps\_config\_s, 375  
pps\_width\_adj  
    vtss\_phy\_ts\_pps\_config\_s, 375  
pr  
    vtss\_phy\_10g\_fifo\_sync\_t, 229  
prbs  
    vtss\_phy\_10g\_lb\_conf\_t, 235  
prbs\_check\_input\_invert  
    vtss\_phy\_10g\_prbs\_mon\_conf\_t, 278  
prbs\_gen  
    vtss\_phy\_10g\_pcs\_prbs\_gen\_conf\_t, 267  
prbs\_inv  
    vtss\_phy\_10g\_ib\_conf\_t, 235  
prbs\_mon  
    vtss\_phy\_10g\_pcs\_prbs\_mon\_conf\_t, 267  
prbsn\_sel  
    vtss\_phy\_10g\_prbs\_mon\_conf\_t, 278  
prbsn\_tx\_io  
    vtss\_phy\_10g\_prbs\_gen\_conf\_t, 276  
prbsn\_tx\_iw  
    vtss\_phy\_10g\_prbs\_gen\_conf\_t, 276  
prbsn\_tx\_sel  
    vtss\_phy\_10g\_prbs\_gen\_conf\_t, 275  
pre\_cb  
    vtss\_fdma\_tx\_info\_t, 132  
pre\_cb\_ctxt1  
    vtss\_fdma\_tx\_info\_t, 132  
pre\_cb\_ctxt2  
    vtss\_fdma\_tx\_info\_t, 132  
prefix\_size  
    vtss\_ip\_network\_t, 142  
    vtss\_ipv4\_network\_t, 143  
    vtss\_ipv6\_network\_t, 145  
prev\_version  
    vtss\_restart\_status\_t, 462  
prio  
    vtss\_fdma\_ch\_cfg\_t, 129  
    vtss\_qce\_action\_t, 432  
    vtss\_vstax\_tx\_header\_t, 528  
prio\_enable  
    vtss\_qce\_action\_t, 432  
prios  
    vtss\_qos\_conf\_t, 448  
prop  
    vtss\_port\_counters\_t, 405  
proto  
    vtss\_ace\_frame\_ipv4\_t, 39  
    vtss\_ace\_frame\_ipv6\_t, 43  
    vtss\_qce\_frame\_ipv4\_t, 436  
    vtss\_qce\_frame\_ipv6\_t, 438  
    vtss\_vce\_frame\_ipv4\_t, 501  
    vtss\_vce\_frame\_ipv6\_t, 503  
proto\_id  
    vtss\_phy\_ts\_ach\_conf\_t, 335  
ptp  
    vtss\_ace\_frame\_etype\_t, 37  
    vtss\_ace\_frame\_ipv4\_t, 42  
    vtss\_ace\_frame\_ipv6\_t, 46  
    vtss\_acl\_action\_t, 59  
    vtss\_phy\_10g\_pkt\_gen\_conf\_t, 269  
    vtss\_phy\_ts\_engine\_flow\_conf\_t, 341  
ptp\_action  
    vtss\_acl\_action\_t, 59  
    vtss\_packet\_tx\_info\_t, 201  
ptp\_conf  
    vtss\_phy\_ts\_engine\_action\_t, 339  
    vtss\_phy\_ts\_ptp\_engine\_action\_t, 378  
ptp\_id  
    vtss\_packet\_tx\_info\_t, 201  
ptp\_timestamp  
    vtss\_packet\_tx\_info\_t, 202  
ptp\_ts\_ns  
    vtss\_phy\_10g\_pkt\_gen\_conf\_t, 271  
ptp\_ts\_sec  
    vtss\_phy\_10g\_pkt\_gen\_conf\_t, 270  
pvrid  
    vtss\_vlan\_port\_conf\_t, 514  
qos\_class\_map  
    vtss\_qos\_port\_conf\_t, 455  
qsgmii  
    vtss\_serdes\_macro\_conf\_t, 466  
queue  
    vtss\_packet\_rx\_conf\_t, 170

queue\_mask  
     vtss\_packet\_rx\_header\_t, 172

queue\_no  
     vtss\_vstax\_tx\_header\_t, 529

queue\_pct  
     vtss\_qos\_port\_conf\_t, 458

r  
     vtss\_vcap\_vr\_t, 498

r\_ctrl  
     vtss\_phy\_10g\_ob\_status\_t, 264

range  
     vtss\_phy\_ts\_eth\_conf\_t, 346  
     vtss\_phy\_ts\_ptp\_conf\_t, 377  
     vtss\_phy\_ts\_y1731\_oam\_conf\_t, 386

range\_en  
     vtss\_phy\_ts\_ptp\_conf\_t, 376  
     vtss\_phy\_ts\_y1731\_oam\_conf\_t, 385

rate  
     vtss\_acl\_policer\_conf\_t, 60  
     vtss\_phy\_10g\_mode\_t, 260  
     vtss\_policer\_t, 396  
     vtss\_shaper\_t, 471

raw\_ident  
     vtss\_irq\_status\_t, 149

raw\_mask  
     vtss\_irq\_status\_t, 149

raw\_status  
     vtss\_irq\_status\_t, 149

rcomp  
     vtss\_phy\_10g\_serdes\_status\_t, 284

rcvrd\_clk  
     vtss\_phy\_10g\_mode\_t, 255

rcvrd\_clk\_div  
     vtss\_phy\_10g\_mode\_t, 255

rdil  
     vtss\_ewis\_cons\_act\_s, 85

rdil\_on\_ais\_l  
     vtss\_ewis\_rdil\_cons\_act\_s, 103

rdil\_on\_lof  
     vtss\_ewis\_rdil\_cons\_act\_s, 103

rdil\_on\_lopc  
     vtss\_ewis\_rdil\_cons\_act\_s, 103

rdil\_on\_los  
     vtss\_ewis\_rdil\_cons\_act\_s, 103

recvrd\_clk\_sel  
     vtss\_phy\_10g\_host\_clk\_conf\_t, 232  
     vtss\_phy\_10g\_line\_clk\_conf\_t, 250

red\_v3  
     vtss\_qos\_conf\_t, 452

reg  
     vtss\_packet\_rx\_conf\_t, 171

reg\_read  
     vtss\_init\_conf\_t, 136

reg\_write  
     vtss\_init\_conf\_t, 136

remote\_entry  
     vtss\_mac\_table\_entry\_t, 158

remote\_fault  
     vtss\_phy\_statistic\_t, 329

vtss\_phy\_10g\_clause\_37\_adv\_t, 222  
     vtss\_phy\_media\_serdes\_pcs\_CNTL\_t, 321  
     vtss\_port\_clause\_37\_adv\_t, 398  
     vtss\_port\_status\_t, 430

replaced  
     vtss\_mac\_table\_status\_t, 159

req  
     vtss\_ace\_frame\_arp\_t, 34

request\_10g  
     vtss\_phy\_10g\_kr\_status\_aneg\_t, 244

request\_1g  
     vtss\_phy\_10g\_kr\_status\_aneg\_t, 244

request\_fec\_change  
     vtss\_phy\_10g\_kr\_status\_aneg\_t, 244

reset  
     vtss\_phy\_10g\_apc\_status\_t, 207  
     vtss\_phy\_10g\_pkt\_mon\_conf\_t, 272

response  
     vtss\_acl\_ptp\_action\_conf\_t, 62

restart  
     vtss\_phy\_mac\_serdes\_pcs\_CNTL\_t, 318  
     vtss\_restart\_status\_t, 461

restart\_info\_port  
     vtss\_init\_conf\_t, 138

restart\_info\_src  
     vtss\_init\_conf\_t, 138

revision  
     vtss\_chip\_id\_t, 68  
     vtss\_phy\_10g\_id\_t, 239  
     vtss\_phy\_type\_t, 387

rgmii  
     vtss\_phy\_reset\_conf\_t, 326

rgmii\_skew\_delay\_psec\_t  
     vtss\_phy\_api.h, 900

rmon  
     vtss\_port\_counters\_t, 405

route  
     vtss\_routing\_entry\_t, 463

rx\_alloc\_cb  
     vtss\_fdma\_cfg\_t, 123

rx\_allow\_multiple\_dcbs  
     vtss\_fdma\_cfg\_t, 124

rx\_allow\_vlan\_tag\_mismatch  
     vtss\_fdma\_cfg\_t, 124

rx\_buf\_cnt  
     vtss\_fdma\_cfg\_t, 122

rx\_cb  
     vtss\_fdma\_cfg\_t, 125

rx\_ch  
     vtss\_phy\_10g\_lane\_sync\_conf\_t, 249

rx\_clk\_skew\_ps  
     vtss\_phy\_rgmii\_conf\_t, 328

rx\_dont\_reinsert\_vlan\_tag  
     vtss\_fdma\_cfg\_t, 124

rx\_dont\_strip\_vlan\_tag  
     vtss\_fdma\_cfg\_t, 124

rx\_err\_cnt\_base\_tx  
     vtss\_phy\_statistic\_t, 329

rx\_etherStatsBroadcastPkts  
    vtss\_port\_rmon\_counters\_t, 421

rx\_etherStatsCRCAlignErrors  
    vtss\_port\_rmon\_counters\_t, 421

rx\_etherStatsDropEvents  
    vtss\_port\_rmon\_counters\_t, 420

rx\_etherStatsFragments  
    vtss\_port\_rmon\_counters\_t, 422

rx\_etherStatsJabbers  
    vtss\_port\_rmon\_counters\_t, 422

rx\_etherStatsMulticastPkts  
    vtss\_port\_rmon\_counters\_t, 421

rx\_etherStatsOctets  
    vtss\_port\_rmon\_counters\_t, 420

rx\_etherStatsOversizePkts  
    vtss\_port\_rmon\_counters\_t, 422

rx\_etherStatsPkts  
    vtss\_port\_rmon\_counters\_t, 421

rx\_etherStatsPkts1024to1518Octets  
    vtss\_port\_rmon\_counters\_t, 423

rx\_etherStatsPkts128to255Octets  
    vtss\_port\_rmon\_counters\_t, 423

rx\_etherStatsPkts1519toMaxOctets  
    vtss\_port\_rmon\_counters\_t, 423

rx\_etherStatsPkts256to511Octets  
    vtss\_port\_rmon\_counters\_t, 423

rx\_etherStatsPkts64Octets  
    vtss\_port\_rmon\_counters\_t, 422

rx\_etherStatsPkts65to127Octets  
    vtss\_port\_rmon\_counters\_t, 422

rx\_etherStatsUndersizePkts  
    vtss\_port\_rmon\_counters\_t, 421

rx\_fault  
    vtss\_sublayer\_status\_t, 472

rx\_frames  
    vtss\_basic\_counters\_t, 67

rx\_info  
    tag\_vtss\_fdma\_list, 32

rx\_link  
    vtss\_sublayer\_status\_t, 472

rx\_macro  
    vtss\_phy\_10g\_lane\_sync\_conf\_t, 248

rx\_mtu  
    vtss\_fdma\_cfg\_t, 122

rx\_prio  
    vtss\_port\_proprietary\_counters\_t, 419

rx\_ts\_len  
    vtss\_phy\_ts\_init\_conf\_t, 359

rx\_ts\_pos  
    vtss\_phy\_ts\_init\_conf\_t, 358

s\_etype  
    vtss\_vlan\_conf\_t, 514

s\_tag  
    vtss\_qce\_tag\_t, 447

    vtss\_vce\_tag\_t, 511

sampling\_rate

    vtss\_sflow\_port\_conf\_t, 467

save  
    vtss\_ts\_alt\_clock\_mode\_t, 480

scan\_conf  
    vtss\_phy\_10g\_vscope\_scan\_status\_t, 299

scan\_type  
    vtss\_phy\_10g\_vscope\_conf\_t, 296

sd6g\_calib\_done  
    vtss\_phy\_10g\_mode\_t, 263

sd\_active\_high  
    vtss\_port\_conf\_t, 401

sd\_enable  
    vtss\_port\_conf\_t, 401

sd\_internal  
    vtss\_port\_conf\_t, 401

sec  
    vtss\_timeofday\_t, 477

sec\_msb  
    vtss\_timestamp\_t, 478

seconds  
    vtss\_phy\_timestamp\_t, 333

    vtss\_timestamp\_t, 478

section\_cons\_act  
    vtss\_ewis\_conf\_s, 82

section\_txти  
    vtss\_ewis\_conf\_s, 82

secure\_on\_enable  
    vtss\_phy\_wol\_conf\_t, 390

select  
    vtss\_eee\_port\_state\_t, 78

seq\_zero  
    vtss\_ace\_frame\_ipv4\_t, 42

    vtss\_ace\_frame\_ipv6\_t, 46

sequence\_id  
    vtss\_phy\_ts\_fifo\_sig\_t, 350

ser\_inv  
    vtss\_phy\_10g\_base\_kr\_conf\_t, 213

ser\_led\_frame\_rate  
    vtss\_phy\_enhanced\_led\_control\_t, 311

ser\_led\_output\_1  
    vtss\_phy\_enhanced\_led\_control\_t, 310

ser\_led\_output\_2  
    vtss\_phy\_enhanced\_led\_control\_t, 311

ser\_led\_select  
    vtss\_phy\_enhanced\_led\_control\_t, 311

serdes  
    vtss\_init\_conf\_t, 139

    vtss\_port\_conf\_t, 404

serdes1g\_vdd  
    vtss\_serdes\_macro\_conf\_t, 465

serdes6g\_vdd  
    vtss\_serdes\_macro\_conf\_t, 465

serdes\_aneg\_ena  
    vtss\_phy\_mac\_serdes\_pcs\_ctrl\_t, 319

serdes\_conf  
    vtss\_phy\_10g\_mode\_t, 259

serdes\_fields\_t, 29

    ob\_post0, 30

ob\_sr, 30  
 serdes\_pol\_inv\_in  
     vtss\_phy\_mac\_serdes\_pcs\_ctrl\_t, 319  
     vtss\_phy\_media\_serdes\_pcs\_ctrl\_t, 321  
 serdes\_pol\_inv\_out  
     vtss\_phy\_mac\_serdes\_pcs\_ctrl\_t, 319  
     vtss\_phy\_media\_serdes\_pcs\_ctrl\_t, 321  
 serdes\_tx\_bad  
     vtss\_phy\_statistic\_t, 329  
 serdes\_tx\_good  
     vtss\_phy\_statistic\_t, 329  
 set\_smac\_to\_port\_mac  
     vtss\_acl\_ptp\_action\_conf\_t, 62  
 sflow\_port\_no  
     vtss\_packet\_rx\_info\_t, 181  
 sflow\_queue  
     vtss\_packet\_rx\_queue\_map\_t, 191  
 sflow\_type  
     vtss\_packet\_rx\_info\_t, 180  
 sfp\_dac  
     vtss\_port\_serdes\_conf\_t, 427  
 sgmii\_in\_pre  
     vtss\_phy\_mac\_serdes\_pcs\_ctrl\_t, 318  
 sgmii\_out\_pre  
     vtss\_phy\_mac\_serdes\_pcs\_ctrl\_t, 319  
 shaper\_port  
     vtss\_qos\_port\_conf\_t, 454  
 shaper\_queue  
     vtss\_qos\_port\_conf\_t, 454  
 sig\_det  
     vtss\_phy\_10g\_ib\_status\_t, 237  
 sig\_mask  
     vtss\_phy\_ts\_fifo\_sig\_t, 350  
 sigdet  
     vtss\_phy\_conf\_t, 307  
 sip  
     vtss\_ace\_frame\_arp\_t, 35  
     vtss\_ace\_frame\_ipv4\_t, 39  
     vtss\_ace\_frame\_ipv6\_t, 43  
     vtss\_ace\_sip\_smac\_t, 51  
     vtss\_qce\_frame\_ipv4\_t, 436  
     vtss\_qce\_frame\_ipv6\_t, 438  
     vtss\_vce\_frame\_ipv4\_t, 502  
     vtss\_vce\_frame\_ipv6\_t, 503  
 sip\_dip\_enable  
     vtss\_aggr\_mode\_t, 64  
 sip\_eq\_dip  
     vtss\_ace\_frame\_ipv4\_t, 41  
     vtss\_ace\_frame\_ipv6\_t, 46  
 sip\_smac  
     vtss\_ace\_frame\_ipv4\_t, 42  
 size  
     vtss\_packet\_rx\_queue\_conf\_t, 188  
 skip\_rev\_check  
     vtss\_phy\_10g\_fifo\_sync\_t, 229  
     vtss\_phy\_ts\_fifo\_conf\_t, 349  
 slew  
     vtss\_phy\_10g\_ob\_status\_t, 264  
 slewrate  
     vtss\_phy\_10g\_base\_kr\_conf\_t, 213  
 sm  
     vtss\_phy\_10g\_kr\_status\_aneg\_t, 244  
 smac  
     vtss\_ace\_frame\_arp\_t, 34  
     vtss\_ace\_frame\_etype\_t, 37  
     vtss\_ace\_frame\_llc\_t, 47  
     vtss\_ace\_frame\_snap\_t, 49  
     vtss\_ace\_sip\_smac\_t, 51  
     vtss\_phy\_10g\_pkt\_gen\_conf\_t, 270  
     vtss\_port\_flow\_control\_conf\_t, 410  
     vtss\_qce\_mac\_t, 444  
     vtss\_vce\_mac\_t, 508  
 smac\_enable  
     vtss\_aggr\_mode\_t, 63  
 smac\_match  
     vtss\_ace\_frame\_arp\_t, 34  
 snap  
     vtss\_ace\_frame\_snap\_t, 49  
     vtss\_ace\_t, 54  
     vtss\_qce\_key\_t, 442  
     vtss\_vce\_key\_t, 507  
 snd\_lvl\_after\_top  
     vtss\_phy\_ts\_mpls\_conf\_t, 367  
 snd\_lvl\_before\_end  
     vtss\_phy\_ts\_mpls\_conf\_t, 368  
 source  
     vtss\_gpio\_10g\_gpio\_mode\_t, 134  
 sp  
     vtss\_vstax\_rx\_header\_t, 526  
 speed  
     port\_custom\_conf\_t, 27  
     vtss\_phy\_forced\_t, 312  
     vtss\_port\_conf\_t, 402  
     vtss\_port\_status\_t, 430  
 speed\_100M  
     vtss\_port\_sgmii\_aneg\_t, 428  
 speed\_100m\_fdx  
     vtss\_phy\_aneg\_t, 302  
 speed\_100m\_hdx  
     vtss\_phy\_aneg\_t, 301  
 speed\_10M  
     vtss\_port\_sgmii\_aneg\_t, 428  
 speed\_10m\_fdx  
     vtss\_phy\_aneg\_t, 301  
 speed\_10m\_hdx  
     vtss\_phy\_aneg\_t, 301  
 speed\_1G  
     vtss\_port\_sgmii\_aneg\_t, 428  
 speed\_1g\_fdx  
     vtss\_phy\_aneg\_t, 302  
 speed\_1g\_hdx  
     vtss\_phy\_aneg\_t, 302  
 spi\_32bit\_read\_write  
     vtss\_init\_conf\_t, 137  
 spi\_64bit\_read\_write  
     vtss\_init\_conf\_t, 137

spi\_daisy\_input  
    vtss\_phy\_daisy\_chain\_conf\_t, 308  
spi\_daisy\_output  
    vtss\_phy\_daisy\_chain\_conf\_t, 309  
spi\_read\_write  
    vtss\_init\_conf\_t, 137  
sport  
    vtss\_ace\_frame\_ipv4\_t, 40  
    vtss\_ace\_frame\_ipv6\_t, 44  
    vtss\_qce\_frame\_ipv4\_t, 437  
    vtss\_qce\_frame\_ipv6\_t, 439  
sport\_dport\_enable  
    vtss\_aggr\_mode\_t, 64  
sport\_eq\_dport  
    vtss\_ace\_frame\_ipv4\_t, 42  
    vtss\_ace\_frame\_ipv6\_t, 46  
sport\_mask  
    vtss\_phy\_ts\_ip\_conf\_t, 363  
sport\_val  
    vtss\_phy\_ts\_ip\_conf\_t, 362  
squelch  
    vtss\_phy\_clock\_conf\_t, 304  
squelch\_inv  
    vtss\_phy\_10g\_ckout\_conf\_t, 220  
    vtss\_phy\_10g\_sckout\_conf\_t, 283  
squelch\_on\_lopc  
    vtss\_phy\_10g\_rxckout\_conf\_t, 282  
squelch\_on\_pcs\_fault  
    vtss\_phy\_10g\_rxckout\_conf\_t, 281  
squelsh  
    vtss\_sync\_clock\_in\_t, 473  
srate  
    vtss\_phy\_10g\_pkt\_gen\_conf\_t, 271  
src  
    vtss\_phy\_10g\_ckout\_conf\_t, 220  
    vtss\_phy\_10g\_sckout\_conf\_t, 283  
    vtss\_phy\_clock\_conf\_t, 303  
src\_ip  
    vtss\_phy\_ts\_fifo\_sig\_t, 351  
src\_port\_identity  
    vtss\_phy\_ts\_fifo\_sig\_t, 350  
sref\_clk\_div  
    vtss\_phy\_10g\_mode\_t, 255  
stack\_depth  
    vtss\_phy\_ts\_mpls\_conf\_t, 366  
stack\_level  
    vtss\_phy\_ts\_mpls\_conf\_t, 368  
stack\_port\_a  
    vtss\_vstax\_route\_entry\_t, 524  
stack\_port\_b  
    vtss\_vstax\_route\_entry\_t, 524  
stack\_queue  
    vtss\_packet\_rx\_queue\_map\_t, 191  
stack\_ref\_point  
    vtss\_phy\_ts\_mpls\_conf\_t, 366  
static\_conf  
    vtss\_ewis\_conf\_s, 82  
status  
    vtss\_phy\_10g\_status\_t, 293  
    vtss\_phy\_veriphy\_result\_t, 389  
stripped\_tag  
    vtss\_packet\_rx\_info\_t, 177  
stuck\_at\_01  
    vtss\_phy\_10g\_prbs\_mon\_conf\_t, 280  
stuck\_at\_par  
    vtss\_phy\_10g\_prbs\_mon\_conf\_t, 279  
suspend\_tick\_cnt  
    vtss\_fdma\_throttle\_cfg\_t, 131  
sw\_tstamp  
    tag\_vtss\_fdma\_list, 32  
    vtss\_packet\_rx\_info\_t, 179  
    vtss\_packet\_rx\_meta\_t, 186  
switch\_frm  
    vtss\_packet\_tx\_info\_t, 196  
sym  
    vtss\_packet\_tx\_info\_t, 200  
symmetric\_pause  
    vtss\_phy\_10g\_clause\_37\_adv\_t, 222  
    vtss\_phy\_aneg\_t, 302  
    vtss\_port\_clause\_37\_adv\_t, 398  
TRUE  
    types.h, 578  
table  
    vtss\_vstax\_route\_table\_t, 525  
tag  
    vtss\_mirror\_conf\_t, 161  
    vtss\_packet\_rx\_header\_t, 173  
    vtss\_packet\_rx\_info\_t, 176  
    vtss\_packet\_tx\_info\_t, 197  
    vtss\_qce\_key\_t, 441  
    vtss\_vce\_key\_t, 506  
tag\_class\_enable  
    vtss\_qos\_port\_conf\_t, 455  
tag\_default\_dei  
    vtss\_qos\_port\_conf\_t, 457  
tag\_default\_pcp  
    vtss\_qos\_port\_conf\_t, 457  
tag\_dei\_map  
    vtss\_qos\_port\_conf\_t, 457  
tag\_pcp\_map  
    vtss\_qos\_port\_conf\_t, 457  
tag\_range\_mode  
    vtss\_phy\_ts\_eth\_conf\_t, 345  
tag\_remark\_mode  
    vtss\_qos\_port\_conf\_t, 456  
tag\_type  
    vtss\_packet\_rx\_info\_t, 176  
tag\_vtss\_fdma\_list, 30  
    act\_len, 31  
    alloc\_ptr, 32  
    frm\_ptr, 31  
    next, 33  
    rx\_info, 32  
    sw\_tstamp, 32  
    user, 32  
tagged

vtss\_ace\_vlan\_t, 56  
 vtss\_qce\_tag\_t, 447  
 vtss\_vce\_tag\_t, 511  
 tagprio  
     vtss\_tci\_t, 476  
 target  
     vtss\_inst\_create\_t, 139  
 tbi  
     vtss\_phy\_reset\_conf\_t, 326  
 tc\_op\_mode  
     vtss\_phy\_ts\_init\_conf\_t, 360  
 tci  
     vtss\_vstax\_tx\_header\_t, 528  
 tcp\_ack  
     vtss\_ace\_frame\_ipv4\_t, 41  
     vtss\_ace\_frame\_ipv6\_t, 45  
 tcp\_fin  
     vtss\_ace\_frame\_ipv4\_t, 40  
     vtss\_ace\_frame\_ipv6\_t, 45  
 tcp\_psh  
     vtss\_ace\_frame\_ipv4\_t, 41  
     vtss\_ace\_frame\_ipv6\_t, 45  
 tcp\_rst  
     vtss\_ace\_frame\_ipv4\_t, 41  
     vtss\_ace\_frame\_ipv6\_t, 45  
 tcp\_syn  
     vtss\_ace\_frame\_ipv4\_t, 40  
     vtss\_ace\_frame\_ipv6\_t, 45  
 tcp\_urg  
     vtss\_ace\_frame\_ipv4\_t, 41  
     vtss\_ace\_frame\_ipv6\_t, 46  
 test\_conf  
     vtss\_ewis\_conf\_s, 83  
 test\_pattern\_ana  
     vtss\_ewis\_test\_conf\_s, 110  
 test\_pattern\_gen  
     vtss\_ewis\_test\_conf\_s, 110  
 thrd\_lvl\_after\_top  
     vtss\_phy\_ts\_mpls\_conf\_t, 367  
 thrd\_lvl\_before\_end  
     vtss\_phy\_ts\_mpls\_conf\_t, 368  
 timeout  
     vtss\_mtimer\_t, 162  
 timestamp  
     vtss\_phy\_10g\_timestamp\_val\_t, 295  
 to\_cpu  
     vtss\_policer\_ext\_t, 394  
 tod\_get\_ns\_cnt\_cb\_t  
     vtss\_misc\_api.h, 709  
 top  
     vtss\_phy\_ts\_mpls\_conf\_t, 367  
 top\_down  
     vtss\_phy\_ts\_mpls\_conf\_t, 367  
 topology\_type  
     vtss\_vstax\_route\_table\_t, 525  
 tpid  
     vtss\_packet\_port\_filter\_t, 168  
     vtss\_phy\_ts\_eth\_conf\_t, 343  
                 vtss\_vlan\_tag\_t, 516  
 train  
     vtss\_phy\_10g\_base\_kr\_status\_t, 216  
 training  
     vtss\_phy\_10g\_base\_kr\_train\_aneg\_t, 217  
 trans\_vid  
     vtss\_vlan\_trans\_grp2vlan\_conf\_t, 518  
 trig\_ch\_id  
     vtss\_phy\_10g\_auto\_failover\_conf\_t, 209  
 trmthd\_c0  
     vtss\_phy\_10g\_base\_kr\_training\_t, 219  
 trmthd\_cm  
     vtss\_phy\_10g\_base\_kr\_training\_t, 219  
 trmthd\_cp  
     vtss\_phy\_10g\_base\_kr\_training\_t, 218  
 ts  
     vtss\_ts\_timestamp\_t, 485  
 ts\_fifo\_drop\_cnt  
     vtss\_phy\_ts\_stats\_t, 384  
 ts\_fifo\_tx\_cnt  
     vtss\_phy\_ts\_stats\_t, 384  
 ts\_format  
     vtss\_phy\_ts\_ifet\_mpls\_ach\_oam\_conf\_t, 357  
 ts\_id  
     vtss\_ts\_id\_t, 482  
 ts\_offset  
     vtss\_phy\_ts\_generic\_action\_t, 355  
 ts\_type  
     vtss\_phy\_ts\_generic\_action\_t, 355  
 ts\_valid  
     vtss\_ts\_timestamp\_t, 486  
 tstamp\_id  
     vtss\_packet\_rx\_info\_t, 179  
 tstamp\_id\_decoded  
     vtss\_packet\_rx\_info\_t, 179  
 tstampat\_cnt  
     vtss\_ewis\_test\_status\_s, 111  
 tti  
     vtss\_ewis\_tti\_s, 112  
 ttl  
     vtss\_ace\_frame\_ipv4\_t, 38  
     vtss\_ace\_frame\_ipv6\_t, 44  
     vtss\_vstax\_port\_conf\_t, 523  
     vtss\_vstax\_tx\_header\_t, 528  
 tx\_b1  
     vtss\_ewis\_tx\_passthru\_s, 117  
 tx\_b2  
     vtss\_ewis\_tx\_passthru\_s, 118  
 tx\_buf\_cnt  
     vtss\_fdma\_cfg\_t, 125  
 tx\_c2  
     vtss\_ewis\_tx\_oh\_s, 115  
 tx\_ch  
     vtss\_phy\_10g\_lane\_sync\_conf\_t, 249  
 tx\_clk\_skew\_ps  
     vtss\_phy\_rgmiif\_conf\_t, 328  
 tx\_dcc\_l  
     vtss\_ewis\_tx\_oh\_s, 114

vtss\_ewis\_tx\_passthru\_s, 119  
tx\_dcc\_s  
    vtss\_ewis\_tx\_oh\_s, 113  
    vtss\_ewis\_tx\_passthru\_s, 117  
tx\_done\_cb  
    vtss\_fdma\_cfg\_t, 125  
tx\_e1  
    vtss\_ewis\_tx\_oh\_s, 113  
    vtss\_ewis\_tx\_passthru\_s, 117  
tx\_e2  
    vtss\_ewis\_tx\_oh\_s, 114  
    vtss\_ewis\_tx\_passthru\_s, 119  
tx\_etherStatsBroadcastPkts  
    vtss\_port\_rmon\_counters\_t, 424  
tx\_etherStatsCollisions  
    vtss\_port\_rmon\_counters\_t, 425  
tx\_etherStatsDropEvents  
    vtss\_port\_rmon\_counters\_t, 424  
tx\_etherStatsMulticastPkts  
    vtss\_port\_rmon\_counters\_t, 424  
tx\_etherStatsOctets  
    vtss\_port\_rmon\_counters\_t, 424  
tx\_etherStatsPkts  
    vtss\_port\_rmon\_counters\_t, 424  
tx\_etherStatsPkts1024to1518Octets  
    vtss\_port\_rmon\_counters\_t, 426  
tx\_etherStatsPkts128to255Octets  
    vtss\_port\_rmon\_counters\_t, 425  
tx\_etherStatsPkts1519toMaxOctets  
    vtss\_port\_rmon\_counters\_t, 426  
tx\_etherStatsPkts256to511Octets  
    vtss\_port\_rmon\_counters\_t, 425  
tx\_etherStatsPkts512to1023Octets  
    vtss\_port\_rmon\_counters\_t, 426  
tx\_etherStatsPkts64Octets  
    vtss\_port\_rmon\_counters\_t, 425  
tx\_etherStatsPkts65to127Octets  
    vtss\_port\_rmon\_counters\_t, 425  
tx\_f1  
    vtss\_ewis\_tx\_oh\_s, 113  
    vtss\_ewis\_tx\_passthru\_s, 117  
tx\_f2  
    vtss\_ewis\_tx\_oh\_s, 115  
tx\_fault  
    vtss\_sublayer\_status\_t, 472  
tx\_fifo\_hi\_clk\_cycs  
    vtss\_phy\_ts\_init\_conf\_t, 359  
tx\_fifo\_lo\_clk\_cycs  
    vtss\_phy\_ts\_init\_conf\_t, 360  
tx\_fifo\_mode  
    vtss\_phy\_ts\_init\_conf\_t, 359  
tx\_fifo\_spi\_conf  
    vtss\_phy\_ts\_init\_conf\_t, 359  
tx\_frames  
    vtss\_basic\_counters\_t, 67  
tx\_j0  
    vtss\_ewis\_tx\_passthru\_s, 116  
tx\_k1  
    vtss\_ewis\_tx\_passthru\_s, 118  
tx\_k1\_k2  
    vtss\_ewis\_tx\_oh\_s, 114  
tx\_k2  
    vtss\_ewis\_tx\_passthru\_s, 118  
tx\_loh  
    vtss\_ewis\_tx\_passthru\_s, 119  
tx\_macro  
    vtss\_phy\_10g\_lane\_sync\_conf\_t, 248  
tx\_n1  
    vtss\_ewis\_tx\_oh\_s, 115  
tx\_oh  
    vtss\_ewis\_conf\_s, 83  
tx\_oh\_passthru  
    vtss\_ewis\_conf\_s, 83  
tx\_out\_bytes  
    vtss\_eee\_port\_counter\_t, 77  
tx\_out\_bytes\_get  
    vtss\_eee\_port\_counter\_t, 77  
tx\_prio  
    vtss\_port\_proprietary\_counters\_t, 419  
tx\_reil  
    vtss\_ewis\_tx\_passthru\_s, 118  
tx\_remote\_fault  
    vtss\_phy\_aneg\_t, 303  
tx\_s1  
    vtss\_ewis\_tx\_oh\_s, 114  
    vtss\_ewis\_tx\_passthru\_s, 119  
tx\_soh  
    vtss\_ewis\_tx\_passthru\_s, 118  
tx\_ts\_len  
    vtss\_phy\_ts\_init\_conf\_t, 359  
tx\_tw  
    vtss\_eee\_port\_conf\_t, 75  
tx\_vstax\_hdr  
    vtss\_packet\_tx\_info\_t, 199  
tx\_z0  
    vtss\_ewis\_tx\_oh\_s, 114  
    vtss\_ewis\_tx\_passthru\_s, 117  
tx\_z1\_z2  
    vtss\_ewis\_tx\_oh\_s, 115  
    vtss\_ewis\_tx\_passthru\_s, 119  
tx\_z3\_z4  
    vtss\_ewis\_tx\_oh\_s, 115  
type  
    vtss\_ace\_t, 53  
    vtss\_dlb\_policer\_conf\_t, 73  
    vtss\_eps\_port\_conf\_t, 79  
    vtss\_fan\_conf\_t, 121  
    vtss\_ip\_addr\_t, 141  
    vtss\_phy\_10g\_id\_t, 240  
    vtss\_qce\_key\_t, 442  
    vtss\_routing\_entry\_t, 463  
    vtss\_sflow\_port\_conf\_t, 466  
    vtss\_vcap\_vr\_t, 497  
    vtss\_vce\_key\_t, 506  
types.h  
    BOOL, 596

FALSE, 578  
 i16, 595  
 i32, 595  
 i64, 595  
 i8, 595  
 MAC\_ADDR\_BROADCAST, 586  
 PRli64, 576  
 PRlu64, 576  
 PRlx64, 576  
 TRUE, 578  
 u16, 596  
 u32, 596  
 u64, 596  
 u8, 595  
 uintptr\_t, 596  
 VTSS\_ACL\_POLICER\_NO\_END, 591  
 VTSS\_ACL\_POLICER\_NO\_START, 590  
 VTSS\_ACL\_POLICERS, 590  
 VTSS\_ACL\_POLICIES, 591  
 VTSS\_ACL\_POLICY\_NO\_END, 592  
 VTSS\_ACL\_POLICY\_NO\_MAX, 591  
 VTSS\_ACL\_POLICY\_NO\_MIN, 591  
 VTSS\_ACL\_POLICY\_NO\_NONE, 591  
 VTSS\_ACL\_POLICY\_NO\_START, 592  
 VTSS\_AGGR\_NO\_END, 587  
 VTSS\_AGGR\_NO\_NONE, 587  
 VTSS\_AGGR\_NO\_START, 587  
 VTSS\_AGGRS, 587  
 VTSS\_BIT64, 576  
 VTSS\_BITMASK64, 577  
 VTSS\_BITRATE\_DISABLED, 584  
 VTSS\_CLOCK\_IDENTITY\_LENGTH, 594  
 VTSS\_DEI\_ARRAY\_SIZE, 583  
 VTSS\_DEI\_END, 583  
 VTSS\_DEI\_START, 583  
 VTSS\_DEIS, 583  
 VTSS\_DPL\_ARRAY\_SIZE, 584  
 VTSS\_DPL\_END, 584  
 VTSS\_DPL\_START, 584  
 VTSS\_DPLS, 583, 584  
 VTSS\_ENCODE\_BITFIELD64, 577  
 VTSS\_ENCODE\_BITMASK64, 577  
 VTSS\_ETYPE\_VTSS, 586  
 VTSS\_EVCS, 586  
 VTSS\_EXTRACT\_BITFIELD64, 577  
 VTSS\_GLAG\_NO\_END, 588  
 VTSS\_GLAG\_NO\_NONE, 588  
 VTSS\_GLAG\_NO\_START, 588  
 VTSS\_GLAG\_PORT\_ARRAY\_SIZE, 589  
 VTSS\_GLAG\_PORT\_END, 589  
 VTSS\_GLAG\_PORT\_START, 588  
 VTSS\_GLAG\_PORTS, 588  
 VTSS\_GLAGS, 587  
 VTSS\_HQOS\_COUNT, 592  
 VTSS\_HQOS\_ID\_NONE, 592  
 VTSS\_INTERVAL\_MS, 593  
 VTSS\_INTERVAL\_NS, 594  
 VTSS\_INTERVAL\_PS, 594  
 VTSS\_INTERVAL\_SEC, 593  
 VTSS\_INTERVAL\_US, 593  
 VTSS\_ISDX\_NONE, 586  
 VTSS\_MAC\_ADDR\_SZ\_BYTES, 586  
 VTSS\_MAX\_TIMEINTERVAL, 593  
 VTSS\_ONE\_MILL, 593  
 VTSS\_ONE\_MIA, 592  
 VTSS\_PACKET\_RATE\_DISABLED, 578  
 VTSS\_PACKET\_RX\_GRP\_CNT, 589  
 VTSS\_PACKET\_RX\_QUEUE\_CNT, 589  
 VTSS\_PACKET\_RX\_QUEUE\_END, 590  
 VTSS\_PACKET\_RX\_QUEUE\_NONE, 590  
 VTSS\_PACKET\_RX\_QUEUE\_START, 590  
 VTSS\_PACKET\_TX\_GRP\_CNT, 589  
 VTSS\_PCP\_ARRAY\_SIZE, 582  
 VTSS\_PCP\_END, 582  
 VTSS\_PCP\_START, 582  
 VTSS\_PCPS, 582  
 VTSS\_PORT\_ARRAY\_SIZE, 580  
 VTSS\_PORT\_COUNT, 578  
 VTSS\_PORT\_IS\_PORT, 580  
 VTSS\_PORT\_NO\_CPU, 579  
 VTSS\_PORT\_NO\_END, 579  
 VTSS\_PORT\_NO\_NONE, 579  
 VTSS\_PORT\_NO\_START, 579  
 VTSS\_PORTS, 579  
 VTSS\_PRIO\_ARRAY\_SIZE, 581  
 VTSS\_PRIO\_END, 581  
 VTSS\_PRIO\_NO\_NONE, 580  
 VTSS\_PRIO\_START, 580  
 VTSS\_PRIOS, 580  
 VTSS\_QUEUE\_ARRAY\_SIZE, 582  
 VTSS\_QUEUE\_END, 581  
 VTSS\_QUEUE\_START, 581  
 VTSS\_QUEUES, 581  
 VTSS\_SEC\_NS\_INTERVAL, 594  
 VTSS\_SYNC\_CLK\_PORT\_ARRAY\_SIZE, 594  
 VTSS\_VID\_ALL, 585  
 VTSS\_VID\_DEFAULT, 585  
 VTSS\_VID\_NULL, 585  
 VTSS\_VID\_RESERVED, 585  
 VTSS\_VIDS, 585  
 vtss\_hqos\_sch\_mode\_t, 604  
 vtss\_ip\_type\_t, 603  
 vtss\_isdx\_t, 597  
 vtss\_mac\_addr\_t, 597  
 vtss\_mem\_flags\_t, 600  
 vtss\_packet\_reg\_type\_t, 602  
 vtss\_packet\_rx\_grp\_t, 597  
 vtss\_packet\_tx\_grp\_t, 597  
 vtss\_policer\_type\_t, 602  
 vtss\_port\_interface\_t, 600  
 vtss\_serdes\_mode\_t, 601  
 vtss\_storm\_policer\_mode\_t, 602  
 vtss\_vcap\_bit\_t, 603  
 vtss\_vcap\_key\_type\_t, 604  
 vtss\_vcap\_vr\_type\_t, 604  
 vtss\_vdd\_t, 603

vtss\_vlan\_frame\_t, 602  
u16  
  types.h, 596  
u32  
  types.h, 596  
u64  
  types.h, 596  
u8  
  types.h, 595  
UNSIGNED\_STORAGE\_COUNT  
  vtss\_phy\_10g\_api.h, 798  
uint  
  vtss\_os\_custom.h, 738  
uintptr\_t  
  types.h, 596  
ulong  
  vtss\_os\_custom.h, 738  
uncorrected\_block\_cnt  
  vtss\_phy\_10g\_kr\_status\_fec\_t, 246  
unidir  
  vtss\_phy\_conf\_t, 307  
unidirectional\_ability  
  vtss\_port\_status\_t, 431  
unknown  
  vtss\_ace\_frame\_arp\_t, 34  
unknown\_broadcast  
  vtss\_policer\_ext\_t, 394  
unknown\_multicast  
  vtss\_policer\_ext\_t, 394  
unknown\_unicast  
  vtss\_policer\_ext\_t, 394  
untagged\_vid  
  vtss\_vlan\_port\_conf\_t, 515  
update  
  vtss\_phy\_10g\_pkt\_mon\_conf\_t, 272  
upper  
  vtss\_phy\_ts\_eth\_conf\_t, 345  
  vtss\_phy\_ts\_mpls\_lvl\_rng\_t, 369  
  vtss\_phy\_ts\_ptp\_conf\_t, 377  
  vtss\_phy\_ts\_y1731\_oam\_conf\_t, 386  
upsid  
  vtss\_mac\_table\_entry\_t, 158  
  vtss\_vstax\_glag\_entry\_t, 522  
  vtss\_vstax\_rx\_header\_t, 526  
  vtss\_vstax\_tx\_header\_t, 528  
upsid\_0  
  vtss\_vstax\_conf\_t, 520  
upsid\_1  
  vtss\_vstax\_conf\_t, 521  
upspn  
  vtss\_mac\_table\_entry\_t, 158  
  vtss\_vstax\_glag\_entry\_t, 522  
usage  
  vtss\_fdma\_ch\_cfg\_t, 127  
use\_as\_intrpt  
  vtss\_gpio\_10g\_gpio\_mode\_t, 135  
use\_conf  
  vtss\_phy\_10g\_mode\_t, 256  
user  
  tag\_vtss\_fdma\_list, 32  
usr\_prio  
  vtss\_ace\_vlan\_t, 56  
  vtss\_qos\_port\_conf\_t, 454  
v  
  vtss\_vcap\_vr\_t, 498  
v3  
  vtss\_phy\_10g\_ob\_status\_t, 265  
v4  
  vtss\_phy\_10g\_ob\_status\_t, 265  
v5  
  vtss\_phy\_10g\_ob\_status\_t, 266  
v\_gpio  
  vtss\_phy\_10g\_auto\_failover\_conf\_t, 209  
VIPER\_B\_FIFO\_RESET  
  options.h, 551  
VTSS\_10G\_PHY\_GPIO\_MAX  
  vtss\_phy\_10g\_api.h, 799  
VTSS\_10G\_PHY\_GPIO\_MAX  
  vtss\_phy\_10g\_api.h, 799  
VTSS\_ACE\_ID\_LAST  
  vtss\_security\_api.h, 1050  
VTSS\_ACL\_POLICER\_NO\_END  
  types.h, 591  
VTSS\_ACL\_POLICER\_NO\_START  
  types.h, 590  
VTSS\_ACL\_POLICERS  
  types.h, 590  
VTSS\_ACL\_POLICIES  
  types.h, 591  
VTSS\_ACL\_POLICY\_NO\_END  
  types.h, 592  
VTSS\_ACL\_POLICY\_NO\_MAX  
  types.h, 591  
VTSS\_ACL\_POLICY\_NO\_MIN  
  types.h, 591  
VTSS\_ACL\_POLICY\_NO\_NONE  
  types.h, 591  
VTSS\_ACL\_POLICY\_NO\_START  
  types.h, 592  
VTSS\_AFI\_FPS\_MAX  
  vtss\_fdma\_api.h, 612  
VTSS\_AGGR\_NO\_END  
  types.h, 587  
VTSS\_AGGR\_NO\_NONE  
  types.h, 587  
VTSS\_AGGR\_NO\_START  
  types.h, 587  
VTSS\_AGGRS  
  types.h, 587  
VTSS\_ARCH\_JAGUAR\_2  
  options.h, 534  
VTSS\_ARCH\_JAGUAR\_2\_B  
  options.h, 534  
VTSS\_ARCH\_MALIBU\_B  
  options.h, 553  
VTSS\_ARCH\_MALIBU

options.h, 553  
**VTSS\_ARCH\_VENICE\_C**  
 options.h, 553  
**VTSS\_BIT64**  
 types.h, 576  
**VTSS\_BITMASK64**  
 types.h, 577  
**VTSS\_BITRATE\_DISABLED**  
 types.h, 584  
**VTSS\_CHIP\_10G\_PHY**  
 options.h, 534  
**VTSS\_CHIP CU PHY**  
 options.h, 551  
**VTSS\_CLOCK\_IDENTITY\_LENGTH**  
 types.h, 594  
**VTSS\_DEI\_ARRAY\_SIZE**  
 types.h, 583  
**VTSS\_DEI\_END**  
 types.h, 583  
**VTSS\_DEI\_START**  
 types.h, 583  
**VTSS\_DEIS**  
 types.h, 583  
**VTSS\_DIV64**  
 vtss\_os\_custom.h, 739  
 vtss\_os\_ecos.h, 744  
 vtss\_os\_linux.h, 756  
**VTSS\_DPL\_ARRAY\_SIZE**  
 types.h, 584  
**VTSS\_DPL\_END**  
 types.h, 584  
**VTSS\_DPL\_START**  
 types.h, 584  
**VTSS\_DPLS**  
 types.h, 583, 584  
**VTSS\_ENCODE\_BITFIELD64**  
 types.h, 577  
**VTSS\_ENCODE\_BITMASK64**  
 types.h, 577  
**VTSS\_ERPI\_ARRAY\_SIZE**  
 vtss\_l2\_api.h, 653  
**VTSS\_ERPI\_END**  
 vtss\_l2\_api.h, 653  
**VTSS\_ERPI\_START**  
 vtss\_l2\_api.h, 652  
**VTSS\_ERPIS**  
 vtss\_l2\_api.h, 652  
**VTSSETYPE\_VTSS**  
 types.h, 586  
**VTSS\_EVCS**  
 types.h, 586  
**VTSS\_EWIS\_AISL\_EV**  
 vtss\_wis\_api.h, 1089  
**VTSS\_EWIS\_AISP\_EV**  
 vtss\_wis\_api.h, 1090  
**VTSS\_EWIS\_B1\_NZ\_EV**  
 vtss\_wis\_api.h, 1092  
**VTSS\_EWIS\_B1\_THRESH\_EV**  
 vtss\_wis\_api.h, 1093  
**VTSS\_EWIS\_B2\_NZ\_EV**  
 vtss\_wis\_api.h, 1092  
**VTSS\_EWIS\_B2\_THRESH\_EV**  
 vtss\_wis\_api.h, 1093  
**VTSS\_EWIS\_B3\_NZ\_EV**  
 vtss\_wis\_api.h, 1093  
**VTSS\_EWIS\_B3\_THRESH\_EV**  
 vtss\_wis\_api.h, 1094  
**VTSS\_EWIS\_FAIS\_EV**  
 vtss\_wis\_api.h, 1088  
**VTSS\_EWIS\_FERDIP\_EV**  
 vtss\_wis\_api.h, 1091  
**VTSS\_EWIS\_FEUNEQP\_EV**  
 vtss\_wis\_api.h, 1091  
**VTSS\_EWIS\_FPLM\_EV**  
 vtss\_wis\_api.h, 1088  
**VTSS\_EWIS\_HIGH\_BER\_EV**  
 vtss\_wis\_api.h, 1092  
**VTSS\_EWIS\_LCDP\_EV**  
 vtss\_wis\_api.h, 1089  
**VTSS\_EWIS\_LOF\_EV**  
 vtss\_wis\_api.h, 1088  
**VTSS\_EWIS\_LOPC\_EV**  
 vtss\_wis\_api.h, 1091  
**VTSS\_EWIS\_LOPP\_EV**  
 vtss\_wis\_api.h, 1090  
**VTSS\_EWIS\_LOS\_EV**  
 vtss\_wis\_api.h, 1089  
**VTSS\_EWIS\_MODULE\_EV**  
 vtss\_wis\_api.h, 1090  
**VTSS\_EWIS\_PCS\_RECEIVE\_FAULT\_PEND**  
 vtss\_wis\_api.h, 1092  
**VTSS\_EWIS\_PLMP\_EV**  
 vtss\_wis\_api.h, 1089  
**VTSS\_EWIS\_RDIL\_EV**  
 vtss\_wis\_api.h, 1089  
**VTSS\_EWIS\_REIL\_EV**  
 vtss\_wis\_api.h, 1091  
**VTSS\_EWIS\_REIL\_NZ\_EV**  
 vtss\_wis\_api.h, 1093  
**VTSS\_EWIS\_REIL\_THRESH\_EV**  
 vtss\_wis\_api.h, 1094  
**VTSS\_EWIS\_REIP\_EV**  
 vtss\_wis\_api.h, 1092  
**VTSS\_EWIS\_REIP\_NZ\_EV**  
 vtss\_wis\_api.h, 1093  
**VTSS\_EWIS\_REIP\_THRESH\_EV**  
 vtss\_wis\_api.h, 1094  
**VTSS\_EWIS\_RXLOL\_EV**  
 vtss\_wis\_api.h, 1090  
**VTSS\_EWIS\_SEF\_EV**  
 vtss\_wis\_api.h, 1088  
**VTSS\_EWIS\_TXLOL\_EV**  
 vtss\_wis\_api.h, 1090  
**VTSS\_EWIS\_UNEQP\_EV**  
 vtss\_wis\_api.h, 1091  
**VTSS\_EXTRACT\_BITFIELD64**

types.h, 577  
VTSS\_FDMA\_CH\_CNT  
    vtss\_fdma\_api.h, 609  
VTSS\_FDMA\_DCB\_SIZE\_BYTES  
    vtss\_fdma\_api.h, 610  
VTSS\_FDMA\_HDR\_SIZE\_BYTES  
    vtss\_fdma\_api.h, 610  
VTSS\_FDMA\_MAX\_DATA\_PER\_DCB\_BYTES  
    vtss\_fdma\_api.h, 610  
VTSS\_FDMA\_MAX\_FRAME\_SIZE\_BYTES  
    vtss\_fdma\_api.h, 611  
VTSS\_FDMA\_MIN\_DATA\_PER\_INJ\_SOF\_DCB\_BY←  
    TES  
    vtss\_fdma\_api.h, 610  
VTSS\_FDMA\_MIN\_DATA\_PER\_NON\_SOF\_DCB\_B←  
    YTES  
    vtss\_fdma\_api.h, 611  
VTSS\_FDMA\_MIN\_DATA\_PER\_XTR\_SOF\_DCB\_B←  
    YTES  
    vtss\_fdma\_api.h, 611  
VTSS\_FDMA\_MIN\_FRAME\_SIZE\_BYTES  
    vtss\_fdma\_api.h, 610  
VTSS\_FEATURE\_10GBASE\_KR  
    options.h, 549  
VTSS\_FEATURE\_10G  
    options.h, 550  
VTSS\_FEATURE\_ACL\_V2  
    options.h, 546  
VTSS\_FEATURE\_ACL  
    options.h, 546  
VTSS\_FEATURE\_AGGR\_GLAG  
    options.h, 537  
VTSS\_FEATURE\_CLAUSE\_37  
    options.h, 535  
VTSS\_FEATURE\_EDC\_FW\_LOAD  
    options.h, 552  
VTSS\_FEATURE\_EEE  
    options.h, 545  
VTSS\_FEATURE\_EXC\_COL\_CONT  
    options.h, 535  
VTSS\_FEATURE\_FAN  
    options.h, 545  
VTSS\_FEATURE\_FDMA  
    options.h, 550  
VTSS\_FEATURE\_INTERRUPTS  
    options.h, 548  
VTSS\_FEATURE\_IRQ\_CONTROL  
    options.h, 549  
VTSS\_FEATURE\_LAYER2  
    options.h, 544  
VTSS\_FEATURE\_LED\_POW\_REDUC  
    options.h, 548  
VTSS\_FEATURE\_MAC\_AGE\_AUTO  
    options.h, 545  
VTSS\_FEATURE\_MAC\_CPU\_QUEUE  
    options.h, 545  
VTSS\_FEATURE\_MISC  
    options.h, 534  
VTSS\_FEATURE\_NPI  
    options.h, 548  
VTSS\_FEATURE\_PACKET\_GROUPING  
    options.h, 544  
VTSS\_FEATURE\_PACKET\_PORT\_REG  
    options.h, 544  
VTSS\_FEATURE\_PACKET\_RX  
    options.h, 543  
VTSS\_FEATURE\_PACKET\_TX  
    options.h, 543  
VTSS\_FEATURE\_PACKET  
    options.h, 543  
VTSS\_FEATURE\_PFC  
    options.h, 537  
VTSS\_FEATURE\_PHY\_TIMESTAMP  
    options.h, 547  
VTSS\_FEATURE\_PORT\_CNT\_BRIDGE  
    options.h, 536  
VTSS\_FEATURE\_PORT\_CNT\_ETHER\_LIKE  
    options.h, 536  
VTSS\_FEATURE\_PORT\_CONTROL  
    options.h, 535  
VTSS\_FEATURE\_PORT\_IFH  
    options.h, 535  
VTSS\_FEATURE\_PORT\_MUX  
    options.h, 537  
VTSS\_FEATURE\_PVLAN  
    options.h, 544  
VTSS\_FEATURE\_QCL\_KEY\_DIP  
    options.h, 538  
VTSS\_FEATURE\_QCL\_KEY\_DMAC  
    options.h, 538  
VTSS\_FEATURE\_QCL\_KEY\_INNER\_TAG  
    options.h, 538  
VTSS\_FEATURE\_QCL\_KEY\_S\_TAG  
    options.h, 538  
VTSS\_FEATURE\_QCL\_PCP\_DEI\_ACTION  
    options.h, 538  
VTSS\_FEATURE\_QCL\_POLICY\_ACTION  
    options.h, 539  
VTSS\_FEATURE\_QCL\_V2  
    options.h, 537  
VTSS\_FEATURE\_QCL  
    options.h, 537  
VTSS\_FEATURE\_QOS\_CLASSIFICATION\_V2  
    options.h, 541  
VTSS\_FEATURE\_QOS\_DSCP\_CLASS\_DP\_AWARE  
    options.h, 542  
VTSS\_FEATURE\_QOS\_DSCP\_REMARK\_V2  
    options.h, 542  
VTSS\_FEATURE\_QOS\_DSCP\_REMARK  
    options.h, 542  
VTSS\_FEATURE\_QOS\_EGRESS\_QUEUE\_SHAPE←  
    RS  
    options.h, 542  
VTSS\_FEATURE\_QOS\_EGRESS\_SHAPERS\_DLB  
    options.h, 542  
VTSS\_FEATURE\_QOS\_POLICER\_BC\_SWITCH

options.h, 539  
**VTSS\_FEATURE\_QOS\_POLICER\_DL**  
 options.h, 543  
**VTSS\_FEATURE\_QOS\_POLICER\_MC\_SWITCH**  
 options.h, 539  
**VTSS\_FEATURE\_QOS\_POLICER\_UC\_SWITCH**  
 options.h, 539  
**VTSS\_FEATURE\_QOS\_PORT\_POLICER\_EXT\_DPBL**  
 options.h, 540  
**VTSS\_FEATURE\_QOS\_PORT\_POLICER\_EXT\_FPS**  
 options.h, 540  
**VTSS\_FEATURE\_QOS\_PORT\_POLICER\_EXT\_FC**  
 options.h, 540  
**VTSS\_FEATURE\_QOS\_PORT\_POLICER\_EXT\_TT\_M\_V2**  
 options.h, 540  
**VTSS\_FEATURE\_QOS\_PORT\_POLICER\_EXT**  
 options.h, 539  
**VTSS\_FEATURE\_QOS\_QUEUE\_POLICER**  
 options.h, 540  
**VTSS\_FEATURE\_QOS\_QUEUE\_TX**  
 options.h, 541  
**VTSS\_FEATURE\_QOS\_SCHEDULER\_DWRR\_CNT**  
 options.h, 541  
**VTSS\_FEATURE\_QOS\_SCHEDULER\_V2**  
 options.h, 541  
**VTSS\_FEATURE\_QOS\_TAG\_REMARK\_V2**  
 options.h, 541  
**VTSS\_FEATURE\_QOS\_WRED\_V3**  
 options.h, 543  
**VTSS\_FEATURE\_QOS**  
 options.h, 536  
**VTSS\_FEATURE\_SERDES\_MACRO\_SETTINGS**  
 options.h, 549  
**VTSS\_FEATURE\_SERIAL\_GPIO**  
 options.h, 535  
**VTSS\_FEATURE\_SFLOW**  
 options.h, 549  
**VTSS\_FEATURE\_SYNCE\_10G**  
 options.h, 552  
**VTSS\_FEATURE\_SYNCE**  
 options.h, 548  
**VTSS\_FEATURE\_TIMESTAMP\_ASYMMETRY\_COMP**  
 options.h, 548  
**VTSS\_FEATURE\_TIMESTAMP\_LATENCY\_COMP**  
 options.h, 547  
**VTSS\_FEATURE\_TIMESTAMP\_ONE\_STEP**  
 options.h, 547  
**VTSS\_FEATURE\_TIMESTAMP\_ORG\_TIME**  
 options.h, 547  
**VTSS\_FEATURE\_TIMESTAMP\_P2P\_DELAY\_COMP**  
 options.h, 547  
**VTSS\_FEATURE\_TIMESTAMP**  
 options.h, 546  
**VTSS\_FEATURE\_VCAP**  
 options.h, 546  
**VTSS\_FEATURE\_VCL**  
 options.h, 546

**VTSS\_FEATURE\_VLAN\_PORT\_V2**  
 options.h, 544  
**VTSS\_FEATURE\_VLAN\_TRANSLATION**  
 options.h, 549  
**VTSS\_FEATURE\_VLAN\_TX\_TAG**  
 options.h, 545  
**VTSS\_FEATURE\_VSTAX\_V2**  
 options.h, 536  
**VTSS\_FEATURE\_VSTAX**  
 options.h, 536  
**VTSS\_FEATURE\_WARM\_START**  
 options.h, 552, 553  
**VTSS\_FEATURE\_WIS**  
 options.h, 553  
**VTSS\_FRAME\_GAP\_DEFAULT**  
 vtss\_port\_api.h, 1024  
**VTSS\_GLAG\_NO\_END**  
 types.h, 588  
**VTSS\_GLAG\_NO\_NONE**  
 types.h, 588  
**VTSS\_GLAG\_NO\_START**  
 types.h, 588  
**VTSS\_GLAG\_PORT\_ARRAY\_SIZE**  
 types.h, 589  
**VTSS\_GLAG\_PORT\_END**  
 types.h, 589  
**VTSS\_GLAG\_PORT\_START**  
 types.h, 588  
**VTSS\_GLAG\_PORTS**  
 types.h, 588  
**VTSS\_GLAGS**  
 types.h, 587  
**VTSS\_HQOS\_COUNT**  
 types.h, 592  
**VTSS\_HQOS\_ID\_NONE**  
 types.h, 592  
**VTSS\_I2C\_NO\_MULTIPLEXER**  
 vtss\_init\_api.h, 626  
**VTSS\_INTERVAL\_MS**  
 types.h, 593  
**VTSS\_INTERVAL\_NS**  
 types.h, 594  
**VTSS\_INTERVAL\_PS**  
 types.h, 594  
**VTSS\_INTERVAL\_SEC**  
 types.h, 593  
**VTSS\_INTERVAL\_US**  
 types.h, 593  
**VTSS\_ISDX\_NONE**  
 types.h, 586  
**VTSS\_JR1\_PACKET\_HDR\_SIZE\_BYTES**  
 vtss\_packet\_api.h, 763  
**VTSS\_JR1\_RX\_IFH\_SIZE**  
 vtss\_packet\_api.h, 765  
**VTSS\_JR2\_PACKET\_HDR\_SIZE\_BYTES**  
 vtss\_packet\_api.h, 764  
**VTSS\_JR2\_RX\_IFH\_SIZE**  
 vtss\_packet\_api.h, 765

VTSS\_L26\_PACKET\_HDR\_SIZE\_BYTES  
vtss\_packet\_api.h, 764

VTSS\_L26\_RX\_IFH\_SIZE  
vtss\_packet\_api.h, 765

VTSS\_LABS  
vtss\_os\_custom.h, 740  
vtss\_os\_ecos.h, 745  
vtss\_os\_linux.h, 756

VTSS\_LLabs  
vtss\_os\_custom.h, 740  
vtss\_os\_ecos.h, 745  
vtss\_os\_linux.h, 756

VTSS\_MAC\_ADDR\_SZ\_BYTES  
types.h, 586

VTSS\_MAC\_ADDRS  
vtss\_l2\_api.h, 645

VTSS\_MAX\_FRAME\_LENGTH\_MAX  
vtss\_port\_api.h, 1025

VTSS\_MAX\_FRAME\_LENGTH\_STANDARD  
vtss\_port\_api.h, 1025

VTSS\_MAX\_TIMEINTERVAL  
types.h, 593

VTSS\_MOD64  
vtss\_os\_custom.h, 739  
vtss\_os\_ecos.h, 745  
vtss\_os\_linux.h, 756

VTSS\_MSLEEP  
vtss\_os\_custom.h, 738  
vtss\_os\_ecos.h, 743  
vtss\_os\_linux.h, 753

VTSS\_MSTI\_ARRAY\_SIZE  
vtss\_l2\_api.h, 648

VTSS\_MSTI\_END  
vtss\_l2\_api.h, 647

VTSS\_MSTI\_START  
vtss\_l2\_api.h, 647

VTSS\_MSTIS  
vtss\_l2\_api.h, 647

VTSS\_MTIMER\_CANCEL  
vtss\_os\_custom.h, 739  
vtss\_os\_ecos.h, 744  
vtss\_os\_linux.h, 754

VTSS\_MTIMER\_START  
vtss\_os\_custom.h, 739  
vtss\_os\_ecos.h, 744  
vtss\_os\_linux.h, 754

VTSS\_MTIMER\_TIMEOUT  
vtss\_os\_custom.h, 739  
vtss\_os\_ecos.h, 744  
vtss\_os\_linux.h, 754

VTSS\_NSLEEP  
vtss\_os\_ecos.h, 743  
vtss\_os\_linux.h, 753

VTSS\_ONE\_MILL  
types.h, 593

VTSS\_ONE\_MIA  
types.h, 592

VTSS\_OPT\_FDMA\_DEBUG  
options.h, 551

VTSS\_OPT\_FDMA\_IRQ\_CONTEXT  
options.h, 551

VTSS\_OPT\_PCIE\_ACCESS  
options.h, 550

VTSS\_OPT\_TRACE  
options.h, 551

VTSS\_OPT\_VAUI\_EQ\_CTRL  
options.h, 552

VTSS\_OPT\_VCORE\_III  
options.h, 550

VTSS\_OS\_BIG\_ENDIAN  
vtss\_os\_linux.h, 753

VTSS\_OS\_COMPILER\_ATTRIBUTE\_ALIGNED  
vtss\_fdma\_api.h, 611  
vtss\_os\_ecos.h, 747

VTSS\_OS\_Ctz64  
vtss\_os\_custom.h, 740  
vtss\_os\_ecos.h, 746  
vtss\_os\_linux.h, 758

VTSS\_OS\_Ctz  
vtss\_os\_custom.h, 740  
vtss\_os\_ecos.h, 745  
vtss\_os\_linux.h, 757

VTSS\_OS\_DCACHE\_FLUSH  
vtss\_os\_ecos.h, 748

VTSS\_OS\_DCACHE\_INVALIDATE  
vtss\_os\_ecos.h, 748

VTSS\_OS\_DCACHE\_LINE\_SIZE\_BYTES  
vtss\_fdma\_api.h, 611  
vtss\_os\_ecos.h, 747

VTSS\_OS\_FREE  
vtss\_os\_custom.h, 741  
vtss\_os\_ecos.h, 746  
vtss\_os\_linux.h, 758

VTSS\_OS\_INTERRUPT\_DISABLE  
vtss\_os\_ecos.h, 750

VTSS\_OS\_INTERRUPT\_FLAGS  
vtss\_os\_ecos.h, 749

VTSS\_OS\_INTERRUPT\_RESTORE  
vtss\_os\_ecos.h, 750

VTSS\_OS\_MALLOC  
vtss\_os\_custom.h, 741  
vtss\_os\_ecos.h, 746  
vtss\_os\_linux.h, 758

VTSS\_OS\_NTOHL  
vtss\_os\_ecos.h, 748  
vtss\_os\_linux.h, 753

VTSS\_OS\_RAND  
vtss\_os\_custom.h, 741  
vtss\_os\_ecos.h, 747  
vtss\_os\_linux.h, 759

VTSS\_OS\_REORDER\_BARRIER  
vtss\_os\_ecos.h, 747

VTSS\_OS\_SCHEDULER\_FLAGS  
vtss\_os\_ecos.h, 749  
vtss\_os\_linux.h, 755

VTSS\_OS\_SCHEDULER\_LOCK  
vtss\_os\_ecos.h, 747

vtss\_os\_ecos.h, 749  
 vtss\_os\_linux.h, 755  
**VTSS\_OS\_SCHEDULER\_UNLOCK**  
 vtss\_os\_ecos.h, 749  
 vtss\_os\_linux.h, 755  
**VTSS\_OS\_TIMESTAMP\_TYPE**  
 vtss\_misc\_api.h, 709  
**VTSS\_OS\_TIMESTAMP**  
 vtss\_misc\_api.h, 709  
**VTSS\_OS\_VIRT\_TO\_PHYS**  
 vtss\_os\_ecos.h, 748  
**VTSS\_PACKET\_HDR\_SIZE\_BYTES**  
 vtss\_packet\_api.h, 764  
**VTSS\_PACKET\_RATE\_DISABLED**  
 types.h, 578  
**VTSS\_PACKET\_RX\_GRP\_CNT**  
 types.h, 589  
**VTSS\_PACKET\_RX\_QUEUE\_CNT**  
 types.h, 589  
**VTSS\_PACKET\_RX\_QUEUE\_END**  
 types.h, 590  
**VTSS\_PACKET\_RX\_QUEUE\_NONE**  
 types.h, 590  
**VTSS\_PACKET\_RX\_QUEUE\_START**  
 types.h, 590  
**VTSS\_PACKET\_TX\_GRP\_CNT**  
 types.h, 589  
**VTSS\_PACKET\_TX\_IFH\_MAX**  
 vtss\_packet\_api.h, 765  
**VTSS\_PCP\_ARRAY\_SIZE**  
 types.h, 582  
**VTSS\_PCP\_END**  
 types.h, 582  
**VTSS\_PCP\_START**  
 types.h, 582  
**VTSS\_PCPS**  
 types.h, 582  
**VTSS\_PHY\_10G\_FEC\_FIXED\_CNT\_THRESH\_EV**  
 vtss\_phy\_10g\_api.h, 808  
**VTSS\_PHY\_10G\_FEC\_UNFIXED\_CNT\_THRESH\_EV**  
 vtss\_phy\_10g\_api.h, 808  
**VTSS\_PHY\_10G\_FIFO\_SYNC**  
 options.h, 550  
**VTSS\_PHY\_10G\_GPIO\_INT\_AGG0\_EV**  
 vtss\_phy\_10g\_api.h, 809  
**VTSS\_PHY\_10G\_GPIO\_INT\_AGG1\_EV**  
 vtss\_phy\_10g\_api.h, 809  
**VTSS\_PHY\_10G\_GPIO\_INT\_AGG2\_EV**  
 vtss\_phy\_10g\_api.h, 809  
**VTSS\_PHY\_10G\_GPIO\_INT\_AGG3\_EV**  
 vtss\_phy\_10g\_api.h, 809  
**VTSS\_PHY\_10G\_HIGH\_BER\_EV**  
 vtss\_phy\_10g\_api.h, 800  
**VTSS\_PHY\_10G\_HIGHBER\_EV**  
 vtss\_phy\_10g\_api.h, 808  
**VTSS\_PHY\_10G\_HOST\_MAC\_LOCAL\_FAULT\_EV**  
 vtss\_phy\_10g\_api.h, 810  
**VTSS\_PHY\_10G\_HOST\_MAC\_REMOTE\_FAULT\_EV**  
 vtss\_phy\_10g\_api.h, 811  
**VTSS\_PHY\_10G\_LINE\_MAC\_LOCAL\_FAULT\_EV**  
 vtss\_phy\_10g\_api.h, 810  
**VTSS\_PHY\_10G\_LINE\_MAC\_REMOTE\_FAULT\_EV**  
 vtss\_phy\_10g\_api.h, 810  
**VTSS\_PHY\_10G\_LINK\_LOS\_EV**  
 vtss\_phy\_10g\_api.h, 800  
**VTSS\_PHY\_10G\_LOPC\_EV**  
 vtss\_phy\_10g\_api.h, 800  
**VTSS\_PHY\_10G\_MACSEC\_DISABLED**  
 vtss\_phy\_10g\_api.h, 799  
**VTSS\_PHY\_10G\_MACSEC\_KEY\_128**  
 vtss\_phy\_10g\_api.h, 799  
**VTSS\_PHY\_10G\_MODULE\_STAT\_EV**  
 vtss\_phy\_10g\_api.h, 801  
**VTSS\_PHY\_10G\_ONE\_LINE\_ACTIVE**  
 vtss\_phy\_10g\_api.h, 798  
**VTSS\_PHY\_10G\_PCS\_RECEIVE\_FAULT\_EV**  
 vtss\_phy\_10g\_api.h, 801  
**VTSS\_PHY\_10G\_RX\_BLK\_DEC\_CNT\_THRESH\_EV**  
 vtss\_phy\_10g\_api.h, 807  
**VTSS\_PHY\_10G\_RX\_CHAR\_DEC\_CNT\_THRESH\_EV**  
 vtss\_phy\_10g\_api.h, 807  
**VTSS\_PHY\_10G\_RX\_LINK\_STAT\_EV**  
 vtss\_phy\_10g\_api.h, 808, 809  
**VTSS\_PHY\_10G\_RX\_LOL\_EV**  
 vtss\_phy\_10g\_api.h, 800, 806  
**VTSS\_PHY\_10G\_RX\_LOS\_EV**  
 vtss\_phy\_10g\_api.h, 806  
**VTSS\_PHY\_10G\_RX\_SEQ\_CNT\_THRESH\_EV**  
 vtss\_phy\_10g\_api.h, 807  
**VTSS\_PHY\_10G\_TIMESTAMP\_DISABLED**  
 vtss\_phy\_10g\_api.h, 799  
**VTSS\_PHY\_10G\_TX\_BLK\_ENC\_CNT\_THRESH\_EV**  
 vtss\_phy\_10g\_api.h, 807  
**VTSS\_PHY\_10G\_TX\_CHAR\_ENC\_CNT\_THRESH\_EV**  
 vtss\_phy\_10g\_api.h, 807  
**VTSS\_PHY\_10G\_TX\_LOL\_EV**  
 vtss\_phy\_10g\_api.h, 800, 806  
**VTSS\_PHY\_10G\_TX\_SEQ\_CNT\_THRESH\_EV**  
 vtss\_phy\_10g\_api.h, 808  
**VTSS\_PHY\_1G\_HOST\_AUTONEG\_RESTART\_EV**  
 vtss\_phy\_10g\_api.h, 810  
**VTSS\_PHY\_1G\_LINE\_AUTONEG\_RESTART\_EV**  
 vtss\_phy\_10g\_api.h, 810  
**VTSS\_PHY\_ACTIPHY\_PWR**  
 vtss\_phy\_api.h, 887  
**VTSS\_PHY\_EWIS\_AISL\_EV**  
 vtss\_phy\_10g\_api.h, 802  
**VTSS\_PHY\_EWIS\_AISP\_EV**  
 vtss\_phy\_10g\_api.h, 803  
**VTSS\_PHY\_EWIS\_B1\_NZ\_EV**  
 vtss\_phy\_10g\_api.h, 804  
**VTSS\_PHY\_EWIS\_B1\_THRESH\_EV**  
 vtss\_phy\_10g\_api.h, 805  
**VTSS\_PHY\_EWIS\_B2\_NZ\_EV**  
 vtss\_phy\_10g\_api.h, 804

VTSS\_PHY\_EWIS\_B2\_THRESH\_EV  
vtss\_phy\_10g\_api.h, 805

VTSS\_PHY\_EWIS\_B3\_NZ\_EV  
vtss\_phy\_10g\_api.h, 804

VTSS\_PHY\_EWIS\_B3\_THRESH\_EV  
vtss\_phy\_10g\_api.h, 805

VTSS\_PHY\_EWIS\_FAIS\_EV  
vtss\_phy\_10g\_api.h, 801

VTSS\_PHY\_EWIS\_FERDIP\_EV  
vtss\_phy\_10g\_api.h, 803

VTSS\_PHY\_EWIS\_FEUNEQP\_EV  
vtss\_phy\_10g\_api.h, 803

VTSS\_PHY\_EWIS\_FPLM\_EV  
vtss\_phy\_10g\_api.h, 801

VTSS\_PHY\_EWIS\_LCDP\_EV  
vtss\_phy\_10g\_api.h, 802

VTSS\_PHY\_EWIS\_LOF\_EV  
vtss\_phy\_10g\_api.h, 802

VTSS\_PHY\_EWIS\_LOPP\_EV  
vtss\_phy\_10g\_api.h, 803

VTSS\_PHY\_EWIS\_PLMP\_EV  
vtss\_phy\_10g\_api.h, 802

VTSS\_PHY\_EWIS\_RDIL\_EV  
vtss\_phy\_10g\_api.h, 802

VTSS\_PHY\_EWIS\_REIL\_EV  
vtss\_phy\_10g\_api.h, 804

VTSS\_PHY\_EWIS\_REIL\_NZ\_EV  
vtss\_phy\_10g\_api.h, 805

VTSS\_PHY\_EWIS\_REIL\_THRESH\_EV  
vtss\_phy\_10g\_api.h, 806

VTSS\_PHY\_EWIS\_REIP\_EV  
vtss\_phy\_10g\_api.h, 804

VTSS\_PHY\_EWIS\_REIP\_NZ\_EV  
vtss\_phy\_10g\_api.h, 805

VTSS\_PHY\_EWIS\_REIP\_THRESH\_EV  
vtss\_phy\_10g\_api.h, 806

VTSS\_PHY\_EWIS\_SEF\_EV  
vtss\_phy\_10g\_api.h, 801

VTSS\_PHY\_EWIS\_UNEQP\_EV  
vtss\_phy\_10g\_api.h, 803

VTSS\_PHY\_LINK\_AMS\_EV  
vtss\_phy\_api.h, 891

VTSS\_PHY\_LINK\_AUTO\_NEG\_COMPLETE\_EV  
vtss\_phy\_api.h, 892

VTSS\_PHY\_LINK\_AUTO\_NEG\_ERROR\_EV  
vtss\_phy\_api.h, 892

VTSS\_PHY\_LINK\_DOWN\_PWR  
vtss\_phy\_api.h, 887

VTSS\_PHY\_LINK\_EXT\_EEE\_LINKFAIL\_EV  
vtss\_phy\_api.h, 895

VTSS\_PHY\_LINK\_EXT\_EEE\_WAIT\_RX\_TQ\_EV  
vtss\_phy\_api.h, 895

VTSS\_PHY\_LINK\_EXT\_EEE\_WAIT\_TS\_EV  
vtss\_phy\_api.h, 895

VTSS\_PHY\_LINK\_EXT\_EEE\_WAKE\_ERR\_EV  
vtss\_phy\_api.h, 894

VTSS\_PHY\_LINK\_EXT\_MACSEC\_EGRESS\_EV  
vtss\_phy\_api.h, 896

VTSS\_PHY\_LINK\_EXT\_MACSEC\_FC\_BUFF\_EV  
vtss\_phy\_api.h, 896

VTSS\_PHY\_LINK\_EXT\_MACSEC\_HOST\_MAC\_EV  
vtss\_phy\_api.h, 895

VTSS\_PHY\_LINK\_EXT\_MACSEC\_INGRESS\_EV  
vtss\_phy\_api.h, 896

VTSS\_PHY\_LINK\_EXT\_MACSEC\_LINE\_MAC\_EV  
vtss\_phy\_api.h, 896

VTSS\_PHY\_LINK\_EXT\_MEM\_INT\_RING\_EV  
vtss\_phy\_api.h, 896

VTSS\_PHY\_LINK\_EXT\_RR\_SW\_COMPL\_EV  
vtss\_phy\_api.h, 895

VTSS\_PHY\_LINK\_EXTENDED\_REG\_INT\_EV  
vtss\_phy\_api.h, 894

VTSS\_PHY\_LINK\_FALSE\_CARRIER\_INT\_EV  
vtss\_phy\_api.h, 893

VTSS\_PHY\_LINK\_FDX\_STATE\_CHANGE\_EV  
vtss\_phy\_api.h, 892

VTSS\_PHY\_LINK\_FFAIL\_EV  
vtss\_phy\_api.h, 891

VTSS\_PHY\_LINK\_INLINE\_POW\_DEV\_DETECT\_EV  
vtss\_phy\_api.h, 892

VTSS\_PHY\_LINK\_LINK\_SPEED\_DS\_DETECT\_EV  
vtss\_phy\_api.h, 893

VTSS\_PHY\_LINK\_LOS\_EV  
vtss\_phy\_api.h, 891

VTSS\_PHY\_LINK\_MASTER\_SLAVE\_RES\_ERR\_EV  
vtss\_phy\_api.h, 894

VTSS\_PHY\_LINK\_RX\_ER\_INT\_EV  
vtss\_phy\_api.h, 894

VTSS\_PHY\_LINK\_RX\_FIFO\_OVERFLOW\_INT\_EV  
vtss\_phy\_api.h, 893

VTSS\_PHY\_LINK\_SPEED\_STATE\_CHANGE\_EV  
vtss\_phy\_api.h, 892

VTSS\_PHY\_LINK\_SYMBOL\_ERR\_INT\_EV  
vtss\_phy\_api.h, 893

VTSS\_PHY\_LINK\_TX\_FIFO\_OVERFLOW\_INT\_EV  
vtss\_phy\_api.h, 893

VTSS\_PHY\_LINK\_UP\_FULL\_PWR  
vtss\_phy\_api.h, 887

VTSS\_PHY\_LINK\_WAKE\_ON\_LAN\_INT\_EV  
vtss\_phy\_api.h, 894

VTSS\_PHY\_OPT\_VERIPH  
options.h, 552

VTSS\_PHY\_PAGE\_0x2DAF  
vtss\_phy\_api.h, 890

VTSS\_PHY\_PAGE\_1588  
vtss\_phy\_api.h, 889

VTSS\_PHY\_PAGE\_EXTENDED\_2  
vtss\_phy\_api.h, 888

VTSS\_PHY\_PAGE\_EXTENDED\_3  
vtss\_phy\_api.h, 888

VTSS\_PHY\_PAGE\_EXTENDED\_4  
vtss\_phy\_api.h, 889

VTSS\_PHY\_PAGE\_EXTENDED  
vtss\_phy\_api.h, 888

VTSS\_PHY\_PAGE\_GPIO  
vtss\_phy\_api.h, 889

VTSS\_PHY\_PAGE\_MACSEC  
     vtss\_phy\_api.h, 889

VTSS\_PHY\_PAGE\_STANDARD  
     vtss\_phy\_api.h, 888

VTSS\_PHY\_PAGE\_TEST  
     vtss\_phy\_api.h, 889

VTSS\_PHY\_PAGE\_TR  
     vtss\_phy\_api.h, 890

VTSS\_PHY\_POWER\_ACTIPHY\_BIT  
     phy.h, 554  
     vtss\_phy\_api.h, 886

VTSS\_PHY\_POWER\_DYNAMIC\_BIT  
     phy.h, 555  
     vtss\_phy\_api.h, 886

VTSS\_PHY\_RECov\_CLK1  
     vtss\_phy\_api.h, 887

VTSS\_PHY\_RECov\_CLK2  
     vtss\_phy\_api.h, 887

VTSS\_PHY\_RECov\_CLK\_NUM  
     vtss\_phy\_api.h, 888

VTSS\_PHY\_REG\_EXTENDED  
     vtss\_phy\_api.h, 890

VTSS\_PHY\_REG\_GPIO  
     vtss\_phy\_api.h, 890

VTSS\_PHY\_REG\_STANDARD  
     vtss\_phy\_api.h, 890

VTSS\_PHY\_REG\_TEST  
     vtss\_phy\_api.h, 891

VTSS\_PHY\_REG\_TR  
     vtss\_phy\_api.h, 891

VTSS\_PHY\_TS\_8487\_XAUI\_SEL\_0  
     vtss\_phy\_ts\_api.h, 972

VTSS\_PHY\_TS\_8487\_XAUI\_SEL\_1  
     vtss\_phy\_ts\_api.h, 972

VTSS\_PHY\_TS\_DATA\_IN\_RSRVD\_FIELD  
     vtss\_phy\_ts\_api.h, 971

VTSS\_PHY\_TS\_EGR\_DATAPATH\_RESET  
     vtss\_phy\_ts\_api.h, 972

VTSS\_PHY\_TS\_EGR\_ENGINE\_ERR  
     vtss\_phy\_ts\_api.h, 970

VTSS\_PHY\_TS\_EGR\_FIFO\_OVERFLOW  
     vtss\_phy\_ts\_api.h, 971

VTSS\_PHY\_TS\_EGR\_FIFO\_RESET  
     vtss\_phy\_ts\_api.h, 973

VTSS\_PHY\_TS\_EGR\_LTC2\_RESET  
     vtss\_phy\_ts\_api.h, 973

VTSS\_PHY\_TS\_EGR\_RW\_FCS\_ERR  
     vtss\_phy\_ts\_api.h, 970

VTSS\_PHY\_TS\_EGR\_TIMESTAMP\_CAPTURED  
     vtss\_phy\_ts\_api.h, 971

VTSS\_PHY\_TS\_ENG\_FLOW\_VALID\_FOR\_CH0  
     vtss\_phy\_ts\_api.h, 969

VTSS\_PHY\_TS\_ENG\_FLOW\_VALID\_FOR\_CH1  
     vtss\_phy\_ts\_api.h, 969

VTSS\_PHY\_TS\_ETH\_ADDR\_MATCH\_48BIT  
     vtss\_phy\_ts\_api.h, 964

VTSS\_PHY\_TS\_ETH\_ADDR\_MATCH\_ANY\_MULTI-  
     CAST

                vtss\_phy\_ts\_api.h, 965

VTSS\_PHY\_TS\_ETH\_ADDR\_MATCH\_ANY\_UNICAST  
     vtss\_phy\_ts\_api.h, 965

VTSS\_PHY\_TS\_ETH\_MATCH\_DEST\_ADDR  
     vtss\_phy\_ts\_api.h, 965

VTSS\_PHY\_TS\_ETH\_MATCH\_SRC\_ADDR  
     vtss\_phy\_ts\_api.h, 965

VTSS\_PHY\_TS\_ETH\_MATCH\_SRC\_OR\_DEST  
     vtss\_phy\_ts\_api.h, 965

VTSS\_PHY\_TS\_FIFO\_SIG\_DEST\_IP  
     vtss\_phy\_ts\_api.h, 961

VTSS\_PHY\_TS\_FIFO\_SIG\_DEST\_MAC  
     vtss\_phy\_ts\_api.h, 962

VTSS\_PHY\_TS\_FIFO\_SIG\_DOMAIN\_NUM  
     vtss\_phy\_ts\_api.h, 961

VTSS\_PHY\_TS\_FIFO\_SIG\_MSG\_TYPE  
     vtss\_phy\_ts\_api.h, 961

VTSS\_PHY\_TS\_FIFO\_SIG\_SEQ\_ID  
     vtss\_phy\_ts\_api.h, 962

VTSS\_PHY\_TS\_FIFO\_SIG\_SOURCE\_PORT\_ID  
     vtss\_phy\_ts\_api.h, 962

VTSS\_PHY\_TS\_FIFO\_SIG\_SRC\_IP  
     vtss\_phy\_ts\_api.h, 961

VTSS\_PHY\_TS\_INGR\_DATAPATH\_RESET  
     vtss\_phy\_ts\_api.h, 972

VTSS\_PHY\_TS\_INGR\_ENGINE\_ERR  
     vtss\_phy\_ts\_api.h, 970

VTSS\_PHY\_TS\_INGR\_LTC1\_RESET  
     vtss\_phy\_ts\_api.h, 972

VTSS\_PHY\_TS\_INGR\_RW\_FCS\_ERR  
     vtss\_phy\_ts\_api.h, 970

VTSS\_PHY\_TS\_INGR\_RW\_PREAM\_ERR  
     vtss\_phy\_ts\_api.h, 970

VTSS\_PHY\_TS\_IP\_MATCH\_DEST  
     vtss\_phy\_ts\_api.h, 968

VTSS\_PHY\_TS\_IP\_MATCH\_SRC\_OR\_DEST  
     vtss\_phy\_ts\_api.h, 968

VTSS\_PHY\_TS\_IP\_MATCH\_SRC  
     vtss\_phy\_ts\_api.h, 967

VTSS\_PHY\_TS\_IP\_VER\_4  
     vtss\_phy\_ts\_api.h, 967

VTSS\_PHY\_TS\_IP\_VER\_6  
     vtss\_phy\_ts\_api.h, 967

VTSS\_PHY\_TS\_LTC\_LOAD\_SAVE\_NEW\_TOD  
     vtss\_phy\_ts\_api.h, 971

VTSS\_PHY\_TS\_LTC\_NEW\_PPS\_INTRPT  
     vtss\_phy\_ts\_api.h, 971

VTSS\_PHY\_TS\_MPLS\_STACK\_DEPTH\_1  
     vtss\_phy\_ts\_api.h, 968

VTSS\_PHY\_TS\_MPLS\_STACK\_DEPTH\_2  
     vtss\_phy\_ts\_api.h, 968

VTSS\_PHY\_TS\_MPLS\_STACK\_DEPTH\_3  
     vtss\_phy\_ts\_api.h, 968

VTSS\_PHY\_TS\_MPLS\_STACK\_DEPTH\_4  
     vtss\_phy\_ts\_api.h, 969

VTSS\_PHY\_TS\_MPLS\_STACK\_REF\_POINT\_END  
     vtss\_phy\_ts\_api.h, 969

VTSS\_PHY\_TS\_MPLS\_STACK\_REF\_POINT\_TOP

vtss\_phy\_ts\_api.h, 969  
VTSS\_PHY\_TS\_SIG\_DEST\_IP\_LEN  
vtss\_phy\_ts\_api.h, 964  
VTSS\_PHY\_TS\_SIG\_DEST\_MAC\_LEN  
vtss\_phy\_ts\_api.h, 964  
VTSS\_PHY\_TS\_SIG\_DOMAIN\_NUM\_LEN  
vtss\_phy\_ts\_api.h, 963  
VTSS\_PHY\_TS\_SIG\_LEN  
vtss\_phy\_ts\_api.h, 962  
VTSS\_PHY\_TS\_SIG\_MSG\_TYPE\_LEN  
vtss\_phy\_ts\_api.h, 963  
VTSS\_PHY\_TS\_SIG\_SEQ\_ID\_LEN  
vtss\_phy\_ts\_api.h, 963  
VTSS\_PHY\_TS\_SIG\_SEQUENCE\_ID\_LEN  
vtss\_phy\_ts\_api.h, 963  
VTSS\_PHY\_TS\_SIG\_SOURCE\_PORT\_ID\_LEN  
vtss\_phy\_ts\_api.h, 963  
VTSS\_PHY\_TS\_SIG\_SRC\_IP\_LEN  
vtss\_phy\_ts\_api.h, 964  
VTSS\_PHY\_TS\_SIG\_TIME\_STAMP\_LEN  
vtss\_phy\_ts\_api.h, 962  
VTSS\_PHY\_TS\_TAG\_RANGE\_INNER  
vtss\_phy\_ts\_api.h, 967  
VTSS\_PHY\_TS\_TAG\_RANGE\_NONE  
vtss\_phy\_ts\_api.h, 966  
VTSS\_PHY\_TS\_TAG\_RANGE\_OUTER  
vtss\_phy\_ts\_api.h, 967  
VTSS\_PHY\_TS\_TAG\_TYPE\_B  
vtss\_phy\_ts\_api.h, 966  
VTSS\_PHY\_TS\_TAG\_TYPE\_C  
vtss\_phy\_ts\_api.h, 966  
VTSS\_PHY\_TS\_TAG\_TYPE\_I  
vtss\_phy\_ts\_api.h, 966  
VTSS\_PHY\_TS\_TAG\_TYPE\_S  
vtss\_phy\_ts\_api.h, 966  
VTSS\_PORT\_ARRAY\_SIZE  
types.h, 580  
VTSS\_PORT\_COUNT  
types.h, 578  
VTSS\_PORT\_IS\_PORT  
types.h, 580  
VTSS\_PORT\_NO\_CPU  
types.h, 579  
VTSS\_PORT\_NO\_END  
types.h, 579  
VTSS\_PORT\_NO\_NONE  
types.h, 579  
VTSS\_PORT\_NO\_START  
types.h, 579  
VTSS\_PORT\_POLICER\_CPU\_QUEUES  
vtss\_qos\_api.h, 1041  
VTSS\_PORT\_POLICERS  
vtss\_qos\_api.h, 1041  
VTSS\_PORTS  
types.h, 579  
VTSS\_PRIO\_ARRAY\_SIZE  
types.h, 581  
VTSS\_PRIO\_END  
types.h, 581  
VTSS\_PRIO\_NO\_NONE  
types.h, 580  
VTSS\_PRIO\_START  
types.h, 580  
VTSS\_PRIO\_SUPER  
vtss\_packet\_api.h, 763  
VTSS\_PRIOS  
types.h, 580  
VTSS\_PTP\_IP\_1588\_VERSION\_2\_1  
vtss\_phy\_ts\_api.h, 964  
VTSS\_PVLAN\_ARRAY\_SIZE  
vtss\_l2\_api.h, 652  
VTSS\_PVLAN\_NO\_DEFAULT  
vtss\_l2\_api.h, 652  
VTSS\_PVLAN\_NO\_END  
vtss\_l2\_api.h, 652  
VTSS\_PVLAN\_NO\_START  
vtss\_l2\_api.h, 651  
VTSS\_PVLANS  
vtss\_l2\_api.h, 651  
VTSS\_QCE\_ID\_LAST  
vtss\_qos\_api.h, 1042  
VTSS\_QCL\_ARRAY\_SIZE  
vtss\_qos\_api.h, 1042  
VTSS\_QCL\_ID\_END  
vtss\_qos\_api.h, 1042  
VTSS\_QCL\_ID\_START  
vtss\_qos\_api.h, 1041  
VTSS\_QCL\_IDS  
vtss\_qos\_api.h, 1041  
VTSS\_QUEUE\_ARRAY\_SIZE  
types.h, 582  
VTSS\_QUEUE\_END  
types.h, 581  
VTSS\_QUEUE\_START  
types.h, 581  
VTSS\_QUEUES  
types.h, 581  
VTSS\_SEC\_NS\_INTERVAL  
types.h, 594  
VTSS\_SLEWRATE\_120PS  
vtss\_phy\_10g\_api.h, 797  
VTSS\_SLEWRATE\_25PS  
vtss\_phy\_10g\_api.h, 796  
VTSS\_SLEWRATE\_35PS  
vtss\_phy\_10g\_api.h, 797  
VTSS\_SLEWRATE\_55PS  
vtss\_phy\_10g\_api.h, 797  
VTSS\_SLEWRATE\_70PS  
vtss\_phy\_10g\_api.h, 797  
VTSS\_SLEWRATE\_INVALID  
vtss\_phy\_10g\_api.h, 797  
VTSS\_SVL\_PACKET\_HDR\_SIZE\_BYTES  
vtss\_packet\_api.h, 764  
VTSS\_SVL\_RX\_IFH\_SIZE  
vtss\_packet\_api.h, 764  
VTSS\_SYNCE\_CLK\_MAX

vtss\_sync\_api.h, 1060  
**VTSS\_SYNCE\_CLK\_PORT\_ARRAY\_SIZE**  
 types.h, 594  
**VTSS\_SYNCE\_CLK\_A**  
 vtss\_sync\_api.h, 1060  
**VTSS\_SYNCE\_CLK\_B**  
 vtss\_sync\_api.h, 1060  
**VTSS\_TIME\_OF\_DAY**  
 vtss\_os\_ecos.h, 744  
 vtss\_os\_linux.h, 755  
**VTSS\_UPSPN\_CPU**  
 vtss\_l2\_api.h, 647  
**VTSS\_UPSPN\_NONE**  
 vtss\_l2\_api.h, 647  
**VTSS\_VCE\_ID\_LAST**  
 vtss\_l2\_api.h, 649  
**VTSS\_VCL\_ARRAY\_SIZE**  
 vtss\_l2\_api.h, 648  
**VTSS\_VCL\_ID\_END**  
 vtss\_l2\_api.h, 648  
**VTSS\_VCL\_ID\_START**  
 vtss\_l2\_api.h, 648  
**VTSS\_VCL\_IDS**  
 vtss\_l2\_api.h, 648  
**VTSS\_VID\_ALL**  
 types.h, 585  
**VTSS\_VID\_DEFAULT**  
 types.h, 585  
**VTSS\_VID\_NULL**  
 types.h, 585  
**VTSS\_VID\_RESERVED**  
 types.h, 585  
**VTSS\_VIDS**  
 types.h, 585  
**VTSS\_VLAN\_TRANS\_FIRST\_GROUP\_ID**  
 vtss\_l2\_api.h, 649  
**VTSS\_VLAN\_TRANS\_GROUP\_MAX\_CNT**  
 vtss\_l2\_api.h, 649  
**VTSS\_VLAN\_TRANS\_LAST\_GROUP\_ID**  
 vtss\_l2\_api.h, 650  
**VTSS\_VLAN\_TRANS\_MAX\_CNT**  
 vtss\_l2\_api.h, 649  
**VTSS\_VLAN\_TRANS\_MAX\_VLAN\_ID**  
 vtss\_l2\_api.h, 650  
**VTSS\_VLAN\_TRANS\_NULL\_CHECK**  
 vtss\_l2\_api.h, 651  
**VTSS\_VLAN\_TRANS\_NULL\_GROUP\_ID**  
 vtss\_l2\_api.h, 649  
**VTSS\_VLAN\_TRANS\_PORT\_BF\_SIZE**  
 vtss\_l2\_api.h, 651  
**VTSS\_VLAN\_TRANS\_VALID\_GROUP\_CHECK**  
 vtss\_l2\_api.h, 650  
**VTSS\_VLAN\_TRANS\_VALID\_VLAN\_CHECK**  
 vtss\_l2\_api.h, 650  
**VTSS\_VLAN\_TRANS\_VID\_START**  
 vtss\_l2\_api.h, 650  
**VTSS\_VSTAX\_HDR\_SIZE**  
 vtss\_packet\_api.h, 763  
**VTSS\_VSTAX\_TTL\_PORT**  
 vtss\_packet\_api.h, 763  
**VTSS\_VSTAX\_UPSID\_LEGAL**  
 vtss\_l2\_api.h, 646  
**VTSS\_VSTAX\_UPSID\_MAX**  
 vtss\_l2\_api.h, 646  
**VTSS\_VSTAX\_UPSID\_MIN**  
 vtss\_l2\_api.h, 646  
**VTSS\_VSTAX\_UPSID\_START**  
 vtss\_l2\_api.h, 646  
**VTSS\_VSTAX\_UPSID\_UNDEF**  
 vtss\_l2\_api.h, 646  
**VTSS\_VSTAX\_UPSIDS**  
 vtss\_l2\_api.h, 645  
**VTSS\_WRED\_DPL\_CNT**  
 vtss\_qos\_api.h, 1040  
**VTSS\_WRED\_GROUP\_CNT**  
 vtss\_qos\_api.h, 1041  
**val**  
 vtss\_eee\_port\_state\_t, 78  
 vtss\_phy\_conf\_1g\_t, 305  
 vtss\_phy\_ts\_eth\_conf\_t, 346, 347  
 vtss\_phy\_ts\_ptp\_conf\_t, 376  
 vtss\_phy\_ts\_y1731\_oam\_conf\_t, 385  
**valid**  
 vtss\_ewis\_tti\_s, 112  
 vtss\_vstax\_rx\_header\_t, 526  
**value**  
 ib\_par\_cfg, 25  
 vtss\_phy\_ts\_ach\_conf\_t, 334  
 vtss\_phy\_ts\_eth\_conf\_t, 346, 347  
 vtss\_phy\_ts\_ptp\_conf\_t, 377  
 vtss\_phy\_ts\_y1731\_oam\_conf\_t, 386  
 vtss\_sgpi\_port\_data\_t, 470  
 vtss\_vcap\_ip\_t, 487  
 vtss\_vcap\_u128\_t, 488  
 vtss\_vcap\_u16\_t, 489  
 vtss\_vcap\_u24\_t, 490  
 vtss\_vcap\_u32\_t, 491  
 vtss\_vcap\_u40\_t, 492  
 vtss\_vcap\_u48\_t, 493  
 vtss\_vcap\_u8\_t, 494  
 vtss\_vcap\_vid\_t, 496  
 vtss\_vcap\_vr\_t, 497  
**venice\_rev\_a\_los\_detection\_workaround**  
 vtss\_phy\_10g\_mode\_t, 260  
**version**  
 vtss\_phy\_ts\_ach\_conf\_t, 334  
 vtss\_phy\_ts\_oam\_engine\_action\_t, 372  
**vga\_adc\_debug**  
 vtss\_phy\_api.h, 925  
**vid**  
 vtss\_ace\_vlan\_t, 56  
 vtss\_mirror\_conf\_t, 161  
 vtss\_packet\_frame\_info\_t, 166  
 vtss\_packet\_port\_info\_t, 169  
 vtss\_qce\_tag\_t, 446  
 vtss\_tci\_t, 475

vtss\_vce\_action\_t, 499  
vtss\_vce\_tag\_t, 510  
vtss\_vid\_mac\_t, 513  
vtss\_vlan\_tag\_t, 516  
vtss\_vlan\_trans\_grp2vlan\_conf\_t, 517  
vid\_mac  
    vtss\_mac\_table\_entry\_t, 156  
vlan  
    vtss\_ace\_t, 53  
    vtss\_routing\_entry\_t, 463  
vlan\_check  
    vtss\_phy\_ts\_eth\_conf\_t, 344  
vml\_format  
    vtss\_debug\_info\_t, 70  
vp  
    vtss\_phy\_10g\_ob\_status\_t, 265  
vr  
    vtss\_vcap\_vr\_t, 498  
vstax  
    vtss\_packet\_rx\_header\_t, 173  
    vtss\_packet\_rx\_info\_t, 182  
    vtss\_packet\_tx\_info\_t, 201  
vstax2  
    vtss\_mac\_table\_entry\_t, 158  
vtail  
    vtss\_phy\_10g\_jitter\_conf\_t, 242  
vtss\_10g\_phy\_gpio\_t  
    vtss\_phy\_10g\_api.h, 827  
vtss\_32\_cntr\_t  
    vtss\_phy\_10g\_api.h, 811  
vtss\_ace\_add  
    vtss\_security\_api.h, 1057  
vtss\_ace\_bit\_t  
    vtss\_security\_api.h, 1053  
vtss\_ace\_counter\_clear  
    vtss\_security\_api.h, 1058  
vtss\_ace\_counter\_get  
    vtss\_security\_api.h, 1058  
vtss\_ace\_del  
    vtss\_security\_api.h, 1057  
vtss\_ace\_frame\_arp\_t, 33  
    arp, 34  
    dip, 36  
    dmac\_match, 35  
    ethernet, 35  
    ip, 35  
    length, 35  
    req, 34  
    sip, 35  
    smac, 34  
    smac\_match, 34  
    unknown, 34  
vtss\_ace\_frame\_etype\_t, 36  
    data, 37  
    dmac, 36  
    etype, 37  
    ptp, 37  
    smac, 37  
vtss\_ace\_frame\_ipv4\_t, 38  
    data, 40  
    dip, 39  
    dport, 40  
    ds, 39  
    fragment, 38  
    options, 39  
    proto, 39  
    ptp, 42  
    seq\_zero, 42  
    sip, 39  
    sip\_eq\_dip, 41  
    sip\_smac, 42  
    sport, 40  
    sport\_eq\_dport, 42  
    tcp\_ack, 41  
    tcp\_fin, 40  
    tcp\_psh, 41  
    tcp\_RST, 41  
    tcp\_SYN, 40  
    tcp\_URG, 41  
    ttl, 38  
vtss\_ace\_frame\_ipv6\_t, 43  
    data, 44  
    dport, 44  
    ds, 44  
    proto, 43  
    ptp, 46  
    seq\_zero, 46  
    sip, 43  
    sip\_eq\_dip, 46  
    sport, 44  
    sport\_eq\_dport, 46  
    tcp\_ack, 45  
    tcp\_fin, 45  
    tcp\_psh, 45  
    tcp\_RST, 45  
    tcp\_SYN, 45  
    tcp\_URG, 46  
    ttl, 44  
vtss\_ace\_frame\_llc\_t, 47  
    dmac, 47  
    llc, 48  
    smac, 47  
vtss\_ace\_frame\_snap\_t, 48  
    dmac, 48  
    smac, 49  
    snap, 49  
vtss\_ace\_init  
    vtss\_security\_api.h, 1056  
vtss\_ace\_ptp\_t, 49  
    enable, 50  
    header, 50  
vtss\_ace\_sip\_smac\_t, 50  
    enable, 51  
    sip, 51  
    smac, 51  
vtss\_ace\_t, 51

action, 53  
 arp, 54  
 dmac\_bc, 53  
 dmac\_mc, 53  
 etype, 54  
 frame, 55  
 id, 52  
 ipv4, 54  
 ipv6, 55  
 llc, 54  
 policy, 52  
 port\_list, 52  
 snap, 54  
 type, 53  
 vlan, 53  
 vtss\_ace\_type\_t  
     vtss\_security\_api.h, 1052  
 vtss\_ace\_vlan\_t, 55  
     cfi, 56  
     tagged, 56  
     usr\_prio, 56  
     vid, 56  
 vtss\_acl\_action\_t, 57  
     cpu, 57  
     cpu\_once, 57  
     cpu\_queue, 57  
     learn, 58  
     mirror, 59  
     police, 58  
     policer\_no, 58  
     port\_action, 58  
     port\_list, 58  
     ptp, 59  
     ptp\_action, 59  
 vtss\_acl\_policer\_conf\_get  
     vtss\_security\_api.h, 1054  
 vtss\_acl\_policer\_conf\_set  
     vtss\_security\_api.h, 1054  
 vtss\_acl\_policer\_conf\_t, 59  
     bit\_rate, 60  
     bit\_rate\_enable, 60  
     rate, 60  
 vtss\_acl\_port\_action\_t  
     vtss\_security\_api.h, 1051  
 vtss\_acl\_port\_conf\_get  
     vtss\_security\_api.h, 1055  
 vtss\_acl\_port\_conf\_set  
     vtss\_security\_api.h, 1055  
 vtss\_acl\_port\_conf\_t, 61  
     action, 61  
     policy\_no, 61  
 vtss\_acl\_port\_counter\_clear  
     vtss\_security\_api.h, 1056  
 vtss\_acl\_port\_counter\_get  
     vtss\_security\_api.h, 1056  
 vtss\_acl\_ptp\_action\_conf\_t, 61  
     copy\_smac\_to\_dmac, 62  
     dom\_sel, 63  
     log\_message\_interval, 62  
     response, 62  
     set\_smac\_to\_port\_mac, 62  
 vtss\_acl\_ptp\_action\_t  
     vtss\_security\_api.h, 1051  
 vtss\_acl\_ptp\_rsp\_t  
     vtss\_security\_api.h, 1052  
 vtss\_aggr\_glag\_members\_get  
     vtss\_l2\_api.h, 685  
 vtss\_aggr\_mode\_get  
     vtss\_l2\_api.h, 685  
 vtss\_aggr\_mode\_set  
     vtss\_l2\_api.h, 685  
 vtss\_aggr\_mode\_t, 63  
     dmac\_enable, 64  
     sip\_dip\_enable, 64  
     smac\_enable, 63  
     sport\_dport\_enable, 64  
 vtss\_aggr\_port\_members\_get  
     vtss\_l2\_api.h, 684  
 vtss\_aggr\_port\_members\_set  
     vtss\_l2\_api.h, 684  
 vtss\_aneg\_t, 64  
     generate\_pause, 65  
     obey\_pause, 65  
 vtss\_api/include/vtss/api/l2\_types.h, 531  
 vtss\_api/include/vtss/api/options.h, 532  
 vtss\_api/include/vtss/api/phy.h, 554  
 vtss\_api/include/vtss/api/port.h, 556  
 vtss\_api/include/vtss/api/types.h, 569  
 vtss\_api/include/vtss\_ae\_api.h, 605  
 vtss\_api/include/vtss\_afi\_api.h, 605  
 vtss\_api/include/vtss\_aneg\_api.h, 605  
 vtss\_api/include/vtss\_api.h, 606  
 vtss\_api/include/vtss\_evc\_api.h, 606  
 vtss\_api/include/vtss\_fdma\_api.h, 608  
 vtss\_api/include/vtss\_gfp\_api.h, 622  
 vtss\_api/include/vtss\_hqos\_api.h, 622  
 vtss\_api/include/vtss\_i2c\_api.h, 623  
 vtss\_api/include/vtss\_init\_api.h, 623  
 vtss\_api/include/vtss\_l2\_api.h, 637  
 vtss\_api/include/vtss\_l3\_api.h, 702  
 vtss\_api/include/vtss\_mac10g\_api.h, 703  
 vtss\_api/include/vtss\_misc\_api.h, 703  
 vtss\_api/include/vtss\_mpls\_api.h, 736  
 vtss\_api/include/vtss\_oam\_api.h, 737  
 vtss\_api/include/vtss\_oha\_api.h, 737  
 vtss\_api/include/vtss\_os.h, 737  
 vtss\_api/include/vtss\_os\_custom.h, 737  
 vtss\_api/include/vtss\_os\_ecos.h, 742  
 vtss\_api/include/vtss\_os\_linux.h, 752  
 vtss\_api/include/vtss\_otn\_api.h, 759  
 vtss\_api/include/vtss\_packet\_api.h, 759  
 vtss\_api/include/vtss\_pcs\_10gbase\_r\_api.h, 782  
 vtss\_api/include/vtss\_phy\_10g\_api.h, 782  
 vtss\_api/include/vtss\_phy\_api.h, 876  
 vtss\_api/include/vtss\_phy\_ts\_api.h, 952  
 vtss\_api/include/vtss\_port\_api.h, 1022

vtss\_api/include/vtss\_qos\_api.h, 1038  
vtss\_api/include/vtss\_rab\_api.h, 1047  
vtss\_api/include/vtss\_security\_api.h, 1048  
vtss\_api/include/vtss\_sfi4\_api.h, 1058  
vtss\_api/include/vtss\_sync\_api.h, 1059  
vtss\_api/include/vtss\_tfi5\_api.h, 1062  
vtss\_api/include/vtss\_ts\_api.h, 1063  
vtss\_api/include/vtss\_upi\_api.h, 1082  
vtss\_api/include/vtss\_wis\_api.h, 1082  
vtss\_api/include/vtss\_xaui\_api.h, 1113  
vtss\_api/include/vtss\_xfi\_api.h, 1113  
vtss\_api\_lock\_t, 65  
    file, 66  
    function, 66  
    inst, 66  
    line, 66  
vtss\_apvlan\_port\_members\_get  
    vtss\_l2\_api.h, 681  
vtss\_apvlan\_port\_members\_set  
    vtss\_l2\_api.h, 681  
vtss\_auth\_port\_state\_get  
    vtss\_security\_api.h, 1053  
vtss\_auth\_port\_state\_set  
    vtss\_security\_api.h, 1053  
vtss\_auth\_state\_t  
    vtss\_security\_api.h, 1051  
vtss\_basic\_counters\_t, 67  
    rx\_frames, 67  
    tx\_frames, 67  
vtss\_callout\_free  
    vtss\_os\_ecos.h, 751  
vtss\_callout\_lock  
    vtss\_misc\_api.h, 717  
vtss\_callout\_malloc  
    vtss\_os\_ecos.h, 751  
vtss\_callout\_trace\_hex\_dump  
    vtss\_misc\_api.h, 716  
vtss\_callout\_trace\_printf  
    vtss\_misc\_api.h, 715  
vtss\_callout\_unlock  
    vtss\_misc\_api.h, 718  
vtss\_channel\_t  
    vtss\_phy\_10g\_api.h, 818  
vtss\_chip\_id\_get  
    vtss\_misc\_api.h, 720  
vtss\_chip\_id\_t, 68  
    part\_number, 68  
    revision, 68  
vtss\_ckout\_data\_sel\_t  
    vtss\_phy\_10g\_api.h, 819  
vtss\_debug\_group\_t  
    vtss\_misc\_api.h, 711  
vtss\_debug\_info\_get  
    vtss\_misc\_api.h, 716  
vtss\_debug\_info\_print  
    vtss\_misc\_api.h, 717  
vtss\_debug\_info\_t, 69  
    chip\_no, 70  
        clear, 70  
        full, 70  
        group, 69  
        layer, 69  
        port\_list, 70  
        vml\_format, 70  
vtss\_debug\_layer\_t  
    vtss\_misc\_api.h, 711  
vtss\_debug\_lock  
    vtss\_misc\_api.h, 718  
vtss\_debug\_lock\_t, 71  
    chip\_no, 71  
vtss\_debug\_reg\_check\_set  
    vtss\_misc\_api.h, 736  
vtss\_debug\_unlock  
    vtss\_misc\_api.h, 718  
vtss\_dev\_all\_event\_enable  
    vtss\_misc\_api.h, 723  
vtss\_dev\_all\_event\_poll  
    vtss\_misc\_api.h, 722  
vtss\_dgroup\_port\_conf\_get  
    vtss\_l2\_api.h, 681  
vtss\_dgroup\_port\_conf\_set  
    vtss\_l2\_api.h, 682  
vtss\_dgroup\_port\_conf\_t, 72  
    dgroup\_no, 72  
vtss\_dlb\_policer\_conf\_t, 72  
    cbs, 74  
    cf, 73  
    cir, 74  
    cm, 73  
    ebs, 74  
    eir, 74  
    enable, 73  
    line\_rate, 73  
    type, 73  
vtss\_dscp\_emode\_t  
    vtss\_qos\_api.h, 1043  
vtss\_dscp\_mode\_t  
    vtss\_qos\_api.h, 1043  
vtss\_eee\_port\_conf\_set  
    vtss\_misc\_api.h, 734  
vtss\_eee\_port\_conf\_t, 75  
    eee\_ena, 75  
    eee\_fast\_queues, 75  
    lp\_advertisement, 76  
    optimized\_for\_power, 76  
    tx\_tw, 75  
vtss\_eee\_port\_counter\_get  
    vtss\_misc\_api.h, 735  
vtss\_eee\_port\_counter\_t, 76  
    fill\_level, 77  
    fill\_level\_get, 77  
    fill\_level\_thres, 77  
    tx\_out\_bytes, 77  
    tx\_out\_bytes\_get, 77  
vtss\_eee\_port\_state\_set  
    vtss\_misc\_api.h, 735

vtss\_eee\_port\_state\_t, 78  
   select, 78  
   val, 78  
 vtss\_eee\_state\_select\_t  
   vtss\_misc\_api.h, 714  
 vtss\_eps\_port\_conf\_get  
   vtss\_l2\_api.h, 696  
 vtss\_eps\_port\_conf\_set  
   vtss\_l2\_api.h, 696  
 vtss\_eps\_port\_conf\_t, 79  
   port\_no, 79  
   type, 79  
 vtss\_eps\_port\_selector\_get  
   vtss\_l2\_api.h, 697  
 vtss\_eps\_port\_selector\_set  
   vtss\_l2\_api.h, 697  
 vtss\_eps\_port\_type\_t  
   vtss\_l2\_api.h, 655  
 vtss\_eps\_selector\_t  
   vtss\_l2\_api.h, 655  
 vtss\_erps\_port\_state\_get  
   vtss\_l2\_api.h, 699  
 vtss\_erps\_port\_state\_set  
   vtss\_l2\_api.h, 699  
 vtss\_erps\_state\_t  
   vtss\_l2\_api.h, 656  
 vtss\_erps\_vlan\_member\_get  
   vtss\_l2\_api.h, 698  
 vtss\_erps\_vlan\_member\_set  
   vtss\_l2\_api.h, 698  
 vtss\_evc\_api.h  
   vtss\_evc\_policer\_conf\_get, 607  
   vtss\_evc\_policer\_conf\_set, 607  
 vtss\_evc\_policer\_conf\_get  
   vtss\_evc\_api.h, 607  
 vtss\_evc\_policer\_conf\_set  
   vtss\_evc\_api.h, 607  
 vtss\_ewis\_aisl\_cons\_act\_s, 80  
   ais\_on\_lof, 80  
   ais\_on\_los, 80  
 vtss\_ewis\_conf\_s, 81  
   ewis\_cntr\_thresh\_conf, 84  
   ewis\_init\_done, 81  
   ewis\_mode, 82  
   exp\_sl, 83  
   force\_mode, 82  
   path\_txti, 83  
   perf\_mode, 84  
   section\_cons\_act, 82  
   section\_txti, 82  
   static\_conf, 82  
   test\_conf, 83  
   tx\_oh, 83  
   tx\_oh\_passthru, 83  
 vtss\_ewis\_cons\_act\_get  
   vtss\_wis\_api.h, 1103  
 vtss\_ewis\_cons\_act\_s, 84  
   aisl, 85  
       fault, 85  
       rdil, 85  
 vtss\_ewis\_cons\_act\_set  
   vtss\_wis\_api.h, 1103  
 vtss\_ewis\_counter\_get  
   vtss\_wis\_api.h, 1110  
 vtss\_ewis\_counter\_s, 85  
   pf\_ebc\_l, 86  
   pf\_ebc\_p, 86  
   pn\_ebc\_l, 86  
   pn\_ebc\_p, 86  
   pn\_ebc\_s, 86  
 vtss\_ewis\_counter\_threshold\_get  
   vtss\_wis\_api.h, 1111  
 vtss\_ewis\_counter\_threshold\_s, 87  
   f\_ebc\_thr\_l, 88  
   f\_ebc\_thr\_p, 88  
   n\_ebc\_thr\_l, 87  
   n\_ebc\_thr\_p, 88  
   n\_ebc\_thr\_s, 87  
 vtss\_ewis\_counter\_threshold\_set  
   vtss\_wis\_api.h, 1111  
 vtss\_ewis\_defects\_get  
   vtss\_wis\_api.h, 1108  
 vtss\_ewis\_defects\_s, 88  
   dais\_l, 90  
   dais\_p, 90  
   dfais\_p, 91  
   dfplm\_p, 91  
   dfuneq\_p, 92  
   dlcd\_p, 91  
   dlof\_s, 89  
   dlop\_p, 90  
   dlos\_s, 89  
   doof\_s, 89  
   dpilm\_p, 91  
   drdi\_l, 90  
   drdi\_p, 91  
   duneq\_p, 90  
 vtss\_ewis\_event\_enable  
   vtss\_wis\_api.h, 1096  
 vtss\_ewis\_event\_force  
   vtss\_wis\_api.h, 1098  
 vtss\_ewis\_event\_poll  
   vtss\_wis\_api.h, 1097  
 vtss\_ewis\_event\_poll\_without\_mask  
   vtss\_wis\_api.h, 1098  
 vtss\_ewis\_event\_t  
   vtss\_wis\_api.h, 1094  
 vtss\_ewis\_exp\_sl\_set  
   vtss\_wis\_api.h, 1105  
 vtss\_ewis\_fault\_cons\_act\_s, 92  
   fault\_on\_aisl, 94  
   fault\_on\_aisp, 94  
   fault\_on\_feaisp, 93  
   fault\_on\_feplmp, 93  
   fault\_on\_lcdp, 94  
   fault\_on\_lof, 93

fault\_on\_lopp, 94  
fault\_on\_los, 93  
fault\_on\_plmp, 94  
fault\_on\_rdil, 93  
fault\_on\_sef, 93  
vtss\_ewis\_force\_conf\_get  
    vtss\_wis\_api.h, 1099  
vtss\_ewis\_force\_conf\_set  
    vtss\_wis\_api.h, 1099  
vtss\_ewis\_force\_mode\_s, 95  
    line\_rx\_force, 95  
    line\_tx\_force, 95  
    path\_force, 96  
vtss\_ewis\_line\_force\_mode\_s, 96  
    force\_ais, 96  
    force\_rdi, 97  
vtss\_ewis\_line\_tx\_force\_mode\_s, 97  
    force\_ais, 97  
    force\_rdi, 98  
vtss\_ewis\_mode\_get  
    vtss\_wis\_api.h, 1102  
vtss\_ewis\_mode\_set  
    vtss\_wis\_api.h, 1102  
vtss\_ewis\_mode\_t  
    vtss\_wis\_api.h, 1095  
vtss\_ewis\_path\_acti\_get  
    vtss\_wis\_api.h, 1109  
vtss\_ewis\_path\_force\_mode\_s, 98  
    force\_rdi, 99  
    force\_ueeq, 98  
vtss\_ewis\_path\_txti\_get  
    vtss\_wis\_api.h, 1106  
vtss\_ewis\_path\_txti\_set  
    vtss\_wis\_api.h, 1105  
vtss\_ewis\_perf\_cntr\_mode\_t  
    vtss\_wis\_api.h, 1095  
vtss\_ewis\_perf\_get  
    vtss\_wis\_api.h, 1110  
vtss\_ewis\_perf\_mode\_get  
    vtss\_wis\_api.h, 1112  
vtss\_ewis\_perf\_mode\_s, 99  
    pf\_ebc\_mode\_l, 100  
    pf\_ebc\_mode\_p, 100  
    pn\_ebc\_mode\_l, 100  
    pn\_ebc\_mode\_p, 100  
    pn\_ebc\_mode\_s, 99  
vtss\_ewis\_perf\_mode\_set  
    vtss\_wis\_api.h, 1112  
vtss\_ewis\_perf\_s, 101  
    pf\_ebc\_l, 101  
    pf\_ebc\_p, 102  
    pn\_ebc\_l, 101  
    pn\_ebc\_p, 102  
    pn\_ebc\_s, 101  
vtss\_ewis\_prbs31\_err\_inj\_set  
    vtss\_wis\_api.h, 1107  
vtss\_ewis\_prbs31\_err\_inj\_t  
    vtss\_wis\_api.h, 1096  
vtss\_ewis\_rdil\_cons\_act\_s, 102  
    rdil\_on\_ais\_l, 103  
    rdil\_on\_lof, 103  
    rdil\_on\_lopc, 103  
    rdil\_on\_los, 103  
vtss\_ewis\_reset  
    vtss\_wis\_api.h, 1103  
vtss\_ewis\_section\_acti\_get  
    vtss\_wis\_api.h, 1109  
vtss\_ewis\_section\_txti\_get  
    vtss\_wis\_api.h, 1104  
vtss\_ewis\_section\_txti\_set  
    vtss\_wis\_api.h, 1104  
vtss\_ewis\_sl\_conf\_s, 104  
    exsl, 104  
vtss\_ewis\_static\_conf\_get  
    vtss\_wis\_api.h, 1099  
vtss\_ewis\_static\_conf\_s, 104  
    ewis\_cnt\_cfg, 108  
    ewis\_lof\_ctrl1, 107  
    ewis\_lof\_ctrl2, 107  
    ewis\_mode\_ctrl, 106  
    ewis\_pmtick\_ctrl, 108  
    ewis\_rx\_ctrl1, 105  
    ewis\_rx\_err\_frc1, 107  
    ewis\_rx\_frm\_ctrl1, 107  
    ewis\_rx\_frm\_ctrl2, 107  
    ewis\_tx\_a1\_a2, 106  
    ewis\_tx\_c2\_h1, 106  
    ewis\_tx\_h2\_h3, 106  
    ewis\_tx\_z0\_e1, 106  
    ewis\_txctrl1, 105  
    ewis\_txctrl2, 105  
vtss\_ewis\_static\_conf\_t  
    vtss\_wis\_api.h, 1094  
vtss\_ewis\_status\_get  
    vtss\_wis\_api.h, 1109  
vtss\_ewis\_status\_s, 108  
    fault, 109  
    link\_stat, 109  
vtss\_ewis\_test\_conf\_s, 109  
    loopback, 110  
    test\_pattern\_ana, 110  
    test\_pattern\_gen, 110  
vtss\_ewis\_test\_counter\_get  
    vtss\_wis\_api.h, 1108  
vtss\_ewis\_test\_mode\_get  
    vtss\_wis\_api.h, 1107  
vtss\_ewis\_test\_mode\_set  
    vtss\_wis\_api.h, 1106  
vtss\_ewis\_test\_pattern\_s  
    vtss\_wis\_api.h, 1096  
vtss\_ewis\_test\_status\_s, 110  
    ana\_sync, 111  
    tstpat\_cnt, 111  
vtss\_ewis\_tti\_mode\_t  
    vtss\_wis\_api.h, 1095  
vtss\_ewis\_tti\_s, 111

mode, 112  
 tti, 112  
 valid, 112  
**vtss\_ewis\_tx\_oh\_get**  
*vtss\_wis\_api.h*, 1100  
**vtss\_ewis\_tx\_oh\_passthru\_get**  
*vtss\_wis\_api.h*, 1101  
**vtss\_ewis\_tx\_oh\_passthru\_set**  
*vtss\_wis\_api.h*, 1101  
**vtss\_ewis\_tx\_oh\_s**, 112  
 tx\_c2, 115  
 tx\_dcc\_l, 114  
 tx\_dcc\_s, 113  
 tx\_e1, 113  
 tx\_e2, 114  
 tx\_f1, 113  
 tx\_f2, 115  
 tx\_k1\_k2, 114  
 tx\_n1, 115  
 tx\_s1, 114  
 tx\_z0, 114  
 tx\_z1\_z2, 115  
 tx\_z3\_z4, 115  
**vtss\_ewis\_tx\_oh\_set**  
*vtss\_wis\_api.h*, 1100  
**vtss\_ewis\_tx\_passthru\_s**, 116  
 tx\_b1, 117  
 tx\_b2, 118  
 tx\_dcc\_l, 119  
 tx\_dcc\_s, 117  
 tx\_e1, 117  
 tx\_e2, 119  
 tx\_f1, 117  
 tx\_j0, 116  
 tx\_k1, 118  
 tx\_k2, 118  
 tx\_loh, 119  
 tx\_reil, 118  
 tx\_s1, 119  
 tx\_soh, 118  
 tx\_z0, 117  
 tx\_z1\_z2, 119  
**vtss\_fan\_conf\_t**, 120  
 fan\_low\_pol, 120  
 fan\_open\_col, 121  
 fan\_pwm\_freq, 120  
 ppr, 121  
 type, 121  
**vtss\_fan\_controller\_init**  
*vtss\_misc\_api.h*, 734  
**vtss\_fan\_cool\_lvl\_get**  
*vtss\_misc\_api.h*, 734  
**vtss\_fan\_cool\_lvl\_set**  
*vtss\_misc\_api.h*, 733  
**vtss\_fan\_rotation\_get**  
*vtss\_misc\_api.h*, 733  
**vtss\_fdma\_api.h**  
*VTSS\_AFI\_FPS\_MAX*, 612  
 VTSS\_FDMA\_CH\_CNT, 609  
 VTSS\_FDMA\_DCB\_SIZE\_BYTES, 610  
 VTSS\_FDMA\_HDR\_SIZE\_BYTES, 610  
 VTSS\_FDMA\_MAX\_DATA\_PER\_DCB\_BYTES,  
 610  
 VTSS\_FDMA\_MAX\_FRAME\_SIZE\_BYTES, 611  
 VTSS\_FDMA\_MIN\_DATA\_PER\_INJ\_SOF\_DC←  
 B\_BYTES, 610  
 VTSS\_FDMA\_MIN\_DATA\_PER\_NON\_SOF\_D←  
 CB\_BYTES, 611  
 VTSS\_FDMA\_MIN\_DATA\_PER\_XTR\_SOF\_DC←  
 B\_BYTES, 611  
 VTSS\_FDMA\_MIN\_FRAME\_SIZE\_BYTES, 610  
 VTSS\_OS\_COMPILER\_ATTRIBUTE\_ALIGNED,  
 611  
 VTSS\_OS\_DCACHE\_LINE\_SIZE\_BYTES, 611  
**vtss\_fdma\_cfg**, 616  
**vtss\_fdma\_ch\_t**, 612  
**vtss\_fdma\_ch\_usage\_t**, 614  
**vtss\_fdma\_dcb\_get**, 618  
**vtss\_fdma\_dcb\_release**, 616  
**vtss\_fdma\_dcb\_type\_t**, 615  
**vtss\_fdma\_irq\_handler**, 621  
**vtss\_fdma\_list\_t**, 614  
**vtss\_fdma\_stats\_clr**, 620  
**vtss\_fdma\_throttle\_cfg\_get**, 619  
**vtss\_fdma\_throttle\_cfg\_set**, 619  
**vtss\_fdma\_throttle\_tick**, 620  
**vtss\_fdma\_tx**, 617  
**vtss\_fdma\_tx\_info\_init**, 618  
**vtss\_fdma\_uninit**, 615  
**vtss\_fdma\_cfg**  
*vtss\_fdma\_api.h*, 616  
**vtss\_fdma\_cfg\_t**, 121  
 enable, 122  
 rx\_alloc\_cb, 123  
 rx\_allow\_multiple\_dcbs, 124  
 rx\_allow\_vlan\_tag\_mismatch, 124  
 rx\_buf\_cnt, 122  
 rx\_cb, 125  
 rx\_dont\_reinsert\_vlan\_tag, 124  
 rx\_dont\_strip\_vlan\_tag, 124  
 rx\_mtu, 122  
 tx\_buf\_cnt, 125  
 tx\_done\_cb, 125  
**vtss\_fdma\_ch\_cfg\_t**, 126  
 chip\_no, 129  
 inj\_grp\_mask, 127  
 list, 128  
 prio, 129  
 usage, 127  
 xtr\_cb, 128  
 xtr\_grp, 127  
**vtss\_fdma\_ch\_t**  
*vtss\_fdma\_api.h*, 612  
**vtss\_fdma\_ch\_usage\_t**  
*vtss\_fdma\_api.h*, 614  
**vtss\_fdma\_dcb\_get**

vtss\_fdma\_api.h, 618  
vtss\_fdma\_dcb\_release  
    vtss\_fdma\_api.h, 616  
vtss\_fdma\_dcb\_type\_t  
    vtss\_fdma\_api.h, 615  
vtss\_fdma\_irq\_handler  
    vtss\_fdma\_api.h, 621  
vtss\_fdma\_list\_t  
    vtss\_fdma\_api.h, 614  
vtss\_fdma\_stats\_clr  
    vtss\_fdma\_api.h, 620  
vtss\_fdma\_throttle\_cfg\_get  
    vtss\_fdma\_api.h, 619  
vtss\_fdma\_throttle\_cfg\_set  
    vtss\_fdma\_api.h, 619  
vtss\_fdma\_throttle\_cfg\_t, 130  
    byte\_limit\_per\_tick, 131  
    frm\_limit\_per\_tick, 131  
    suspend\_tick\_cnt, 131  
vtss\_fdma\_throttle\_tick  
    vtss\_fdma\_api.h, 620  
vtss\_fdma\_tx  
    vtss\_fdma\_api.h, 617  
vtss\_fdma\_tx\_info\_init  
    vtss\_fdma\_api.h, 618  
vtss\_fdma\_tx\_info\_t, 131  
    pre\_cb, 132  
    pre\_cb\_ctxt1, 132  
    pre\_cb\_ctxt2, 132  
vtss\_fdma\_uninit  
    vtss\_fdma\_api.h, 615  
vtss\_fefi\_mode\_t  
    vtss\_phy\_api.h, 905  
vtss\_fiber\_port\_speed\_t  
    port.h, 568  
vtss\_gpio\_10g\_aggr\_intrpt\_t  
    vtss\_phy\_10g\_api.h, 831  
vtss\_gpio\_10g\_chan\_intrpt\_t  
    vtss\_phy\_10g\_api.h, 830  
vtss\_gpio\_10g\_gpio\_intr\_sgnl\_t  
    vtss\_phy\_10g\_api.h, 828  
vtss\_gpio\_10g\_gpio\_mode\_t, 133  
    aggr\_intrpt, 134  
    c\_intrpt, 134  
    in\_sig, 134  
    input, 133  
    mode, 133  
    p\_gpio, 134  
    p\_gpio\_intrpt, 135  
    port, 133  
    source, 134  
    use\_as\_intrpt, 135  
vtss\_gpio\_10g\_input\_t  
    vtss\_phy\_10g\_api.h, 831  
vtss\_gpio\_10g\_no\_t  
    vtss\_phy\_10g\_api.h, 811  
vtss\_gpio\_direction\_set  
    vtss\_misc\_api.h, 723  
vtss\_gpio\_event\_enable  
    vtss\_misc\_api.h, 725  
vtss\_gpio\_event\_poll  
    vtss\_misc\_api.h, 725  
vtss\_gpio\_mode\_set  
    vtss\_misc\_api.h, 723  
vtss\_gpio\_mode\_t  
    vtss\_misc\_api.h, 712  
vtss\_gpio\_read  
    vtss\_misc\_api.h, 724  
vtss\_gpio\_write  
    vtss\_misc\_api.h, 724  
vtss\_hqos\_sch\_mode\_t  
    types.h, 604  
vtss\_i2c\_read\_t  
    vtss\_init\_api.h, 627  
vtss\_i2c\_write\_t  
    vtss\_init\_api.h, 627  
vtss\_init\_api.h  
    VTSS\_I2C\_NO\_MULTIPLEXER, 626  
    vtss\_i2c\_read\_t, 627  
    vtss\_i2c\_write\_t, 627  
    vtss\_init\_conf\_get, 634  
    vtss\_init\_conf\_set, 635  
    vtss\_inst\_create, 634  
    vtss\_inst\_destroy, 634  
    vtss\_inst\_get, 633  
    vtss\_miim\_read\_t, 629  
    vtss\_miim\_write\_t, 630  
    vtss\_mmd\_read\_inc\_t, 631  
    vtss\_mmd\_read\_t, 630  
    vtss\_mmd\_write\_t, 631  
    vtss\_port\_mux\_mode\_t, 632  
    vtss\_reg\_read\_t, 626  
    vtss\_reg\_write\_t, 626  
    vtss\_restart\_conf\_end, 635  
    vtss\_restart\_conf\_get, 636  
    vtss\_restart\_conf\_set, 636  
    vtss\_restart\_status\_get, 636  
    vtss\_restart\_t, 633  
    vtss\_spi\_32bit\_read\_write\_t, 628  
    vtss\_spi\_64bit\_read\_write\_t, 629  
    vtss\_spi\_read\_write\_t, 628  
    vtss\_target\_type\_t, 632  
vtss\_init\_conf\_get  
    vtss\_init\_api.h, 634  
vtss\_init\_conf\_set  
    vtss\_init\_api.h, 635  
vtss\_init\_conf\_t, 135  
    miim\_read, 136  
    miim\_write, 136  
    mmd\_read, 136  
    mmd\_read\_inc, 137  
    mmd\_write, 137  
    mux\_mode, 138  
    pi, 138  
    reg\_read, 136  
    reg\_write, 136

restart\_info\_port, 138  
 restart\_info\_src, 138  
 serdes, 139  
 spi\_32bit\_read\_write, 137  
 spi\_64bit\_read\_write, 137  
 spi\_read\_write, 137  
 warm\_start\_enable, 138  
 vtss\_inst\_create  
     vtss\_init\_api.h, 634  
 vtss\_inst\_create\_t, 139  
     target, 139  
 vtss\_inst\_destroy  
     vtss\_init\_api.h, 634  
 vtss\_inst\_get  
     vtss\_init\_api.h, 633  
 vtss\_internal\_bw\_t  
     vtss\_port\_api.h, 1026  
 vtss\_intr\_cfg  
     vtss\_misc\_api.h, 728  
 vtss\_intr\_mask\_set  
     vtss\_misc\_api.h, 729  
 vtss\_intr\_pol\_negation  
     vtss\_misc\_api.h, 729  
 vtss\_intr\_status\_get  
     vtss\_misc\_api.h, 729  
 vtss\_intr\_sticky\_clear  
     vtss\_misc\_api.h, 720  
 vtss\_intr\_t, 140  
     link\_change, 140  
 vtss\_ip\_addr\_t, 141  
     addr, 142  
     ipv4, 141  
     ipv6, 141  
     type, 141  
 vtss\_ip\_network\_t, 142  
     address, 142  
     prefix\_size, 142  
 vtss\_ip\_type\_t  
     types.h, 603  
 vtss\_ipv4\_mc\_flood\_members\_get  
     vtss\_l2\_api.h, 694  
 vtss\_ipv4\_mc\_flood\_members\_set  
     vtss\_l2\_api.h, 694  
 vtss\_ipv4\_network\_t, 143  
     address, 143  
     prefix\_size, 143  
 vtss\_ipv4\_uc\_t, 144  
     destination, 144  
     network, 144  
 vtss\_ipv6\_mc\_ctrl\_flood\_get  
     vtss\_l2\_api.h, 695  
 vtss\_ipv6\_mc\_ctrl\_flood\_set  
     vtss\_l2\_api.h, 696  
 vtss\_ipv6\_mc\_flood\_members\_get  
     vtss\_l2\_api.h, 694  
 vtss\_ipv6\_mc\_flood\_members\_set  
     vtss\_l2\_api.h, 695  
 vtss\_ipv6\_network\_t, 145  
     address, 145  
     prefix\_size, 145  
 vtss\_ipv6\_t, 146  
     addr, 146  
 vtss\_ipv6\_uc\_t, 147  
     destination, 147  
     network, 147  
 vtss\_irq\_conf\_get  
     vtss\_misc\_api.h, 730  
 vtss\_irq\_conf\_set  
     vtss\_misc\_api.h, 730  
 vtss\_irq\_conf\_t, 147  
     destination, 148  
     external, 148  
 vtss\_irq\_enable  
     vtss\_misc\_api.h, 731  
 vtss\_irq\_status\_get\_and\_mask  
     vtss\_misc\_api.h, 731  
 vtss\_irq\_status\_t, 148  
     active, 149  
     raw\_ident, 149  
     raw\_mask, 149  
     raw\_status, 149  
 vtss\_irq\_t  
     vtss\_misc\_api.h, 714  
 vtss\_isidx\_t  
     types.h, 597  
 vtss\_isolated\_port\_members\_get  
     vtss\_l2\_api.h, 679  
 vtss\_isolated\_port\_members\_set  
     vtss\_l2\_api.h, 679  
 vtss\_l2\_api.h  
     VTSS\_ERPI\_ARRAY\_SIZE, 653  
     VTSS\_ERPI\_END, 653  
     VTSS\_ERPI\_START, 652  
     VTSS\_ERPIS, 652  
     VTSS\_MAC\_ADDRS, 645  
     VTSS\_MSTI\_ARRAY\_SIZE, 648  
     VTSS\_MSTI\_END, 647  
     VTSS\_MSTI\_START, 647  
     VTSS\_MSTIS, 647  
     VTSS\_PVLAN\_ARRAY\_SIZE, 652  
     VTSS\_PVLAN\_NO\_DEFAULT, 652  
     VTSS\_PVLAN\_NO\_END, 652  
     VTSS\_PVLAN\_NO\_START, 651  
     VTSS\_PVLANS, 651  
     VTSS\_UPSPN\_CPU, 647  
     VTSS\_UPSPN\_NONE, 647  
     VTSS\_VCE\_ID\_LAST, 649  
     VTSS\_VCL\_ARRAY\_SIZE, 648  
     VTSS\_VCL\_ID\_END, 648  
     VTSS\_VCL\_ID\_START, 648  
     VTSS\_VCL\_IDS, 648  
     VTSS\_VLAN\_TRANS\_FIRST\_GROUP\_ID, 649

VTSS\_VLAN\_TRANS\_GROUP\_MAX\_CNT, 649  
VTSS\_VLAN\_TRANS\_LAST\_GROUP\_ID, 650  
VTSS\_VLAN\_TRANS\_MAX\_CNT, 649  
VTSS\_VLAN\_TRANS\_MAX\_VLAN\_ID, 650  
VTSS\_VLAN\_TRANS\_NULL\_CHECK, 651  
VTSS\_VLAN\_TRANS\_NULL\_GROUP\_ID, 649  
VTSS\_VLAN\_TRANS\_PORT\_BF\_SIZE, 651  
VTSS\_VLAN\_TRANS\_VALID\_GROUP\_CHECK,  
    650  
VTSS\_VLAN\_TRANS\_VALID\_VLAN\_CHECK,  
    650  
VTSS\_VLAN\_TRANS\_VID\_START, 650  
VTSS\_VSTAX\_UPSID\_LEGAL, 646  
VTSS\_VSTAX\_UPSID\_MAX, 646  
VTSS\_VSTAX\_UPSID\_MIN, 646  
VTSS\_VSTAX\_UPSID\_START, 646  
VTSS\_VSTAX\_UPSID\_UNDEF, 646  
VTSS\_VSTAX\_UPSIDS, 645  
vtss\_aggr\_glag\_members\_get, 685  
vtss\_aggr\_mode\_get, 685  
vtss\_aggr\_mode\_set, 685  
vtss\_aggr\_port\_members\_get, 684  
vtss\_aggr\_port\_members\_set, 684  
vtss\_apvlan\_port\_members\_get, 681  
vtss\_apvlan\_port\_members\_set, 681  
vtss\_dgroup\_port\_conf\_get, 681  
vtss\_dgroup\_port\_conf\_set, 682  
vtss\_eps\_port\_conf\_get, 696  
vtss\_eps\_port\_conf\_set, 696  
vtss\_eps\_port\_selector\_get, 697  
vtss\_eps\_port\_selector\_set, 697  
vtss\_eps\_port\_type\_t, 655  
vtss\_eps\_selector\_t, 655  
vtss\_erps\_port\_state\_get, 699  
vtss\_erps\_port\_state\_set, 699  
vtss\_erps\_state\_t, 656  
vtss\_erps\_vlan\_member\_get, 698  
vtss\_erps\_vlan\_member\_set, 698  
vtss\_ipv4\_mc\_flood\_members\_get, 694  
vtss\_ipv4\_mc\_flood\_members\_set, 694  
vtss\_ipv6\_mc\_ctrl\_flood\_get, 695  
vtss\_ipv6\_mc\_ctrl\_flood\_set, 696  
vtss\_ipv6\_mc\_flood\_members\_get, 694  
vtss\_ipv6\_mc\_flood\_members\_set, 695  
vtss\_isolated\_port\_members\_get, 679  
vtss\_isolated\_port\_members\_set, 679  
vtss\_isolated\_vlan\_get, 678  
vtss\_isolated\_vlan\_set, 678  
vtss\_learn\_port\_mode\_get, 664  
vtss\_learn\_port\_mode\_set, 665  
vtss\_mac\_table\_add, 656  
vtss\_mac\_table\_age, 660  
vtss\_mac\_table\_age\_time\_get, 659  
vtss\_mac\_table\_age\_time\_set, 659  
vtss\_mac\_table\_del, 658  
vtss\_mac\_table\_flush, 660  
vtss\_mac\_table\_get, 658  
vtss\_mac\_table\_get\_next, 659  
vtss\_mac\_table\_glag\_add, 663  
vtss\_mac\_table\_glag\_flush, 663  
vtss\_mac\_table\_port\_flush, 661  
vtss\_mac\_table\_status\_get, 664  
vtss\_mac\_table\_upsid\_flush, 662  
vtss\_mac\_table\_upsid\_upspn\_flush, 662  
vtss\_mac\_table\_vlan\_age, 660  
vtss\_mac\_table\_vlan\_flush, 661  
vtss\_mac\_table\_vlan\_glag\_flush, 664  
vtss\_mac\_table\_vlan\_port\_flush, 662  
vtss\_mc\_flood\_members\_get, 693  
vtss\_mc\_flood\_members\_set, 693  
vtss\_mirror\_conf\_get, 688  
vtss\_mirror\_conf\_set, 688  
vtss\_mirror\_cpu\_egress\_get, 691  
vtss\_mirror\_cpu\_egress\_set, 692  
vtss\_mirror\_cpu\_ingress\_get, 691  
vtss\_mirror\_cpu\_ingress\_set, 691  
vtss\_mirror\_egress\_ports\_get, 690  
vtss\_mirror\_egress\_ports\_set, 690  
vtss\_mirror\_ingress\_ports\_get, 689  
vtss\_mirror\_ingress\_ports\_set, 689  
vtss\_mirror\_monitor\_port\_get, 688  
vtss\_mirror\_monitor\_port\_set, 689  
vtss\_mirror\_tag\_t, 655  
vtss\_mstp\_port\_msti\_state\_get, 668  
vtss\_mstp\_port\_msti\_state\_set, 668  
vtss\_mstp\_vlan\_msti\_get, 667  
vtss\_mstp\_vlan\_msti\_set, 667  
vtss\_port\_state\_get, 665  
vtss\_port\_state\_set, 666  
vtss\_pvlan\_port\_members\_get, 680  
vtss\_pvlan\_port\_members\_set, 680  
vtss\_sflow\_port\_conf\_get, 682  
vtss\_sflow\_port\_conf\_set, 683  
vtss\_sflow\_sampling\_rate\_convert, 683  
vtss\_stp\_port\_state\_get, 666  
vtss\_stp\_port\_state\_set, 666  
vtss\_stp\_state\_t, 653  
vtss\_uc\_flood\_members\_get, 692  
vtss\_uc\_flood\_members\_set, 693  
vtss\_vce\_add, 675  
vtss\_vce\_del, 675  
vtss\_vce\_init, 675  
vtss\_vce\_type\_t, 654  
vtss\_vcl\_port\_conf\_get, 674  
vtss\_vcl\_port\_conf\_set, 674  
vtss\_vlan\_conf\_get, 669  
vtss\_vlan\_conf\_set, 669  
vtss\_vlan\_port\_conf\_get, 669  
vtss\_vlan\_port\_conf\_set, 671  
vtss\_vlan\_port\_members\_get, 671  
vtss\_vlan\_port\_members\_set, 672  
vtss\_vlan\_port\_type\_t, 654  
vtss\_vlan\_trans\_group\_add, 676  
vtss\_vlan\_trans\_group\_del, 676  
vtss\_vlan\_trans\_group\_get, 677  
vtss\_vlan\_trans\_group\_to\_port\_get, 678

vtss\_vlan\_trans\_group\_to\_port\_set, 677  
 vtss\_vlan\_tx\_tag\_get, 673  
 vtss\_vlan\_tx\_tag\_set, 673  
 vtss\_vlan\_tx\_tag\_t, 654  
 vtss\_vlan\_vid\_conf\_get, 672  
 vtss\_vlan\_vid\_conf\_set, 672  
 vtss\_vstax\_conf\_get, 699  
 vtss\_vstax\_conf\_set, 700  
 vtss\_vstax\_glag\_get, 687  
 vtss\_vstax\_glag\_set, 687  
 vtss\_vstax\_master\_upsid\_get, 701  
 vtss\_vstax\_master\_upsid\_set, 702  
 vtss\_vstax\_port\_conf\_get, 700  
 vtss\_vstax\_port\_conf\_set, 701  
 vtss\_vstax\_topology\_set, 702  
 vtss\_vstax\_topology\_type\_t, 656  
 vtss\_vt\_id\_t, 653  
 vtss\_l3\_counters\_t, 150  
   ipv4uc\_received\_frames, 150  
   ipv4uc\_received\_octets, 150  
   ipv4uc\_transmitted\_frames, 151  
   ipv4uc\_transmitted\_octets, 151  
   ipv6uc\_received\_frames, 151  
   ipv6uc\_received\_octets, 151  
   ipv6uc\_transmitted\_frames, 152  
   ipv6uc\_transmitted\_octets, 151  
 vtss\_lb\_type\_t  
   vtss\_phy\_10g\_api.h, 824  
 vtss\_lcpll\_status\_t, 152  
   cal\_done, 153  
   cal\_error, 153  
   fsm\_lock, 153  
   fsm\_stat, 153  
   gain\_stat, 153  
   lock\_status, 152  
 vtss\_learn\_mode\_t, 154  
   automatic, 154  
   cpu, 154  
   discard, 155  
 vtss\_learn\_port\_mode\_get  
   vtss\_l2\_api.h, 664  
 vtss\_learn\_port\_mode\_set  
   vtss\_l2\_api.h, 665  
 vtss\_mac\_addr\_t  
   types.h, 597  
 vtss\_mac\_t, 155  
   addr, 155  
 vtss\_mac\_table\_add  
   vtss\_l2\_api.h, 656  
 vtss\_mac\_table\_age  
   vtss\_l2\_api.h, 660  
 vtss\_mac\_table\_age\_time\_get  
   vtss\_l2\_api.h, 659  
 vtss\_mac\_table\_age\_time\_set  
   vtss\_l2\_api.h, 659  
 vtss\_mac\_table\_del  
   vtss\_l2\_api.h, 658  
 vtss\_mac\_table\_entry\_t, 156  
     aged, 157  
     copy\_to\_cpu, 157  
     cpu\_queue, 157  
     destination, 157  
     enable, 158  
     locked, 157  
     remote\_entry, 158  
     upsid, 158  
     upspn, 158  
     vid\_mac, 156  
     vstax2, 158  
 vtss\_mac\_table\_flush  
   vtss\_l2\_api.h, 660  
 vtss\_mac\_table\_get  
   vtss\_l2\_api.h, 658  
 vtss\_mac\_table\_get\_next  
   vtss\_l2\_api.h, 659  
 vtss\_mac\_table\_glag\_add  
   vtss\_l2\_api.h, 663  
 vtss\_mac\_table\_glag\_flush  
   vtss\_l2\_api.h, 663  
 vtss\_mac\_table\_port\_flush  
   vtss\_l2\_api.h, 661  
 vtss\_mac\_table\_status\_get  
   vtss\_l2\_api.h, 664  
 vtss\_mac\_table\_status\_t, 159  
   aged, 160  
   learned, 159  
   moved, 160  
   replaced, 159  
 vtss\_mac\_table\_upsid\_flush  
   vtss\_l2\_api.h, 662  
 vtss\_mac\_table\_upsid\_upspn\_flush  
   vtss\_l2\_api.h, 662  
 vtss\_mac\_table\_vlan\_age  
   vtss\_l2\_api.h, 660  
 vtss\_mac\_table\_vlan\_flush  
   vtss\_l2\_api.h, 661  
 vtss\_mac\_table\_vlan\_glag\_flush  
   vtss\_l2\_api.h, 664  
 vtss\_mac\_table\_vlan\_port\_flush  
   vtss\_l2\_api.h, 662  
 vtss\_mc\_flood\_members\_get  
   vtss\_l2\_api.h, 693  
 vtss\_mc\_flood\_members\_set  
   vtss\_l2\_api.h, 693  
 vtss\_mem\_flags\_t  
   types.h, 600  
 vtss\_miim\_controller\_t  
   vtss\_port\_api.h, 1025  
 vtss\_miim\_read  
   vtss\_port\_api.h, 1034  
 vtss\_miim\_read\_t  
   vtss\_init\_api.h, 629  
 vtss\_miim\_write  
   vtss\_port\_api.h, 1034  
 vtss\_miim\_write\_t  
   vtss\_init\_api.h, 630

vtss\_mirror\_conf\_get  
    vtss\_l2\_api.h, 688  
vtss\_mirror\_conf\_set  
    vtss\_l2\_api.h, 688  
vtss\_mirror\_conf\_t, 160  
    dei, 162  
    fwd\_enable, 161  
    pcp, 161  
    port\_no, 161  
    tag, 161  
    vid, 161  
vtss\_mirror\_cpu\_egress\_get  
    vtss\_l2\_api.h, 691  
vtss\_mirror\_cpu\_egress\_set  
    vtss\_l2\_api.h, 692  
vtss\_mirror\_cpu\_ingress\_get  
    vtss\_l2\_api.h, 691  
vtss\_mirror\_cpu\_ingress\_set  
    vtss\_l2\_api.h, 691  
vtss\_mirror\_egress\_ports\_get  
    vtss\_l2\_api.h, 690  
vtss\_mirror\_egress\_ports\_set  
    vtss\_l2\_api.h, 690  
vtss\_mirror\_ingress\_ports\_get  
    vtss\_l2\_api.h, 689  
vtss\_mirror\_ingress\_ports\_set  
    vtss\_l2\_api.h, 689  
vtss\_mirror\_monitor\_port\_get  
    vtss\_l2\_api.h, 688  
vtss\_mirror\_monitor\_port\_set  
    vtss\_l2\_api.h, 689  
vtss\_mirror\_tag\_t  
    vtss\_l2\_api.h, 655  
vtss\_misc\_api.h  
    tod\_get\_ns\_cnt\_cb\_t, 709  
    VTSS\_OS\_TIMESTAMP\_TYPE, 709  
    VTSS\_OS\_TIMESTAMP, 709  
    vtss\_callout\_lock, 717  
    vtss\_callout\_trace\_hex\_dump, 716  
    vtss\_callout\_trace\_printf, 715  
    vtss\_callout\_unlock, 718  
    vtss\_chip\_id\_get, 720  
    vtss\_debug\_group\_t, 711  
    vtss\_debug\_info\_get, 716  
    vtss\_debug\_info\_print, 717  
    vtss\_debug\_layer\_t, 711  
    vtss\_debug\_lock, 718  
    vtss\_debug\_reg\_check\_set, 736  
    vtss\_debug\_unlock, 718  
    vtss\_dev\_all\_event\_enable, 723  
    vtss\_dev\_all\_event\_poll, 722  
    vtss\_eee\_port\_conf\_set, 734  
    vtss\_eee\_port\_counter\_get, 735  
    vtss\_eee\_port\_state\_set, 735  
    vtss\_eee\_state\_select\_t, 714  
    vtss\_fan\_controller\_init, 734  
    vtss\_fan\_cool\_lvl\_get, 734  
    vtss\_fan\_cool\_lvl\_set, 733  
vtss\_fan\_rotation\_get, 733  
vtss\_gpio\_direction\_set, 723  
vtss\_gpio\_event\_enable, 725  
vtss\_gpio\_event\_poll, 725  
vtss\_gpio\_mode\_set, 723  
vtss\_gpio\_mode\_t, 712  
vtss\_gpio\_read, 724  
vtss\_gpio\_write, 724  
vtss\_intr\_cfg, 728  
vtss\_intr\_mask\_set, 729  
vtss\_intr\_pol\_negation, 729  
vtss\_intr\_status\_get, 729  
vtss\_intr\_sticky\_clear, 720  
vtss\_irq\_conf\_get, 730  
vtss\_irq\_conf\_set, 730  
vtss\_irq\_enable, 731  
vtss\_irq\_status\_get\_and\_mask, 731  
vtss\_irq\_t, 714  
vtss\_poll\_1sec, 721  
vtss\_ptp\_event\_enable, 722  
vtss\_ptp\_event\_poll, 721  
vtss\_reg\_read, 719  
vtss\_reg\_write, 719  
vtss\_reg\_write\_masked, 719  
vtss\_sgpiobmode\_t, 713  
vtss\_sgpiocnf\_get, 726  
vtss\_sgpiocnf\_set, 726  
vtss\_sgpiocnf\_enable, 728  
vtss\_sgpiocnf\_poll, 727  
vtss\_sgpiomode\_t, 713  
vtss\_sgpioread, 727  
vtss\_temp\_sensor\_get, 732  
vtss\_temp\_sensor\_init, 732  
vtss\_tod\_get\_ns\_cnt, 731  
vtss\_tod\_set\_ns\_cnt\_cb, 732  
vtss\_trace\_conf\_get, 714  
vtss\_trace\_conf\_set, 715  
vtss\_trace\_group\_t, 710  
vtss\_trace\_layer\_t, 710  
vtss\_trace\_level\_t, 711  
vtss\_mmd\_read  
    vtss\_port\_api.h, 1037  
vtss\_mmd\_read\_inc\_t  
    vtss\_init\_api.h, 631  
vtss\_mmd\_read\_t  
    vtss\_init\_api.h, 630  
vtss\_mmd\_write  
    vtss\_port\_api.h, 1038  
vtss\_mmd\_write\_t  
    vtss\_init\_api.h, 631  
vtss\_mstp\_port\_msti\_state\_get  
    vtss\_l2\_api.h, 668  
vtss\_mstp\_port\_msti\_state\_set  
    vtss\_l2\_api.h, 668  
vtss\_mstp\_vlan\_msti\_get  
    vtss\_l2\_api.h, 667  
vtss\_mstp\_vlan\_msti\_set  
    vtss\_l2\_api.h, 667

vtss\_mtimer\_t, 162  
 now, 163  
 timeout, 162  
 vtss\_os\_custom.h, 742  
 vtss\_os\_ecos.h, 750  
 vtss\_npi\_conf\_get  
 vtss\_packet\_api.h, 769  
 vtss\_npi\_conf\_set  
 vtss\_packet\_api.h, 769  
 vtss\_npi\_conf\_t, 163  
 enable, 163  
 port\_no, 164  
 vtss\_os\_custom.h  
 uint, 738  
 ulong, 738  
 VTSS\_DIV64, 739  
 VTSS\_LABS, 740  
 VTSS\_LLabs, 740  
 VTSS\_MOD64, 739  
 VTSS\_MSLEEP, 738  
 VTSS\_MTIMER\_CANCEL, 739  
 VTSS\_MTIMER\_START, 739  
 VTSS\_MTIMER\_TIMEOUT, 739  
 VTSS\_OS\_CTZ64, 740  
 VTSS\_OS\_CTZ, 740  
 VTSS\_OS\_FREE, 741  
 VTSS\_OS\_MALLOC, 741  
 VTSS\_OS\_RAND, 741  
 vtss\_mtimer\_t, 742  
 vtss\_os\_ecos.h  
 llabs, 750  
 VTSS\_DIV64, 744  
 VTSS\_LABS, 745  
 VTSS\_LLabs, 745  
 VTSS\_MOD64, 745  
 VTSS\_MSLEEP, 743  
 VTSS\_MTIMER\_CANCEL, 744  
 VTSS\_MTIMER\_START, 744  
 VTSS\_MTIMER\_TIMEOUT, 744  
 VTSS\_NSLEEP, 743  
 VTSS\_OS\_COMPILER\_ATTRIBUTE\_ALIGNED, 747  
 VTSS\_OS\_CTZ64, 746  
 VTSS\_OS\_CTZ, 745  
 VTSS\_OS\_DCACHE\_FLUSH, 748  
 VTSS\_OS\_DCACHE\_INVALIDATE, 748  
 VTSS\_OS\_DCACHE\_LINE\_SIZE\_BYTES, 747  
 VTSS\_OS\_FREE, 746  
 VTSS\_OS\_INTERRUPT\_DISABLE, 750  
 VTSS\_OS\_INTERRUPT\_FLAGS, 749  
 VTSS\_OS\_INTERRUPT\_RESTORE, 750  
 VTSS\_OS\_MALLOC, 746  
 VTSS\_OS\_NTOHL, 748  
 VTSS\_OS\_RAND, 747  
 VTSS\_OS\_REORDER\_BARRIER, 747  
 VTSS\_OS\_SCHEDULER\_FLAGS, 749  
 VTSS\_OS\_SCHEDULER\_LOCK, 749  
 VTSS\_OS\_SCHEDULER\_UNLOCK, 749  
 VTSS\_OS\_VIRT\_TO\_PHYS, 748  
 VTSS\_TIME\_OF\_DAY, 744  
 vtss\_callout\_free, 751  
 vtss\_callout\_malloc, 751  
 vtss\_mtimer\_t, 750  
 vtss\_os\_linux.h  
 VTSS\_DIV64, 756  
 VTSS\_LABS, 756  
 VTSS\_LLabs, 756  
 VTSS\_MOD64, 756  
 VTSS\_MSLEEP, 753  
 VTSS\_MTIMER\_CANCEL, 754  
 VTSS\_MTIMER\_START, 754  
 VTSS\_MTIMER\_TIMEOUT, 754  
 VTSS\_NSLEEP, 753  
 VTSS\_OS\_BIG\_ENDIAN, 753  
 VTSS\_OS\_CTZ64, 758  
 VTSS\_OS\_CTZ, 757  
 VTSS\_OS\_FREE, 758  
 VTSS\_OS\_MALLOC, 758  
 VTSS\_OS\_NTOHL, 753  
 VTSS\_OS\_RAND, 759  
 VTSS\_OS\_SCHEDULER\_FLAGS, 755  
 VTSS\_OS\_SCHEDULER\_LOCK, 755  
 VTSS\_OS\_SCHEDULER\_UNLOCK, 755  
 VTSS\_TIME\_OF\_DAY, 755  
 vtss\_os\_timestamp\_t, 164  
 hw\_cnt, 164  
 vtss\_packet\_api.h  
 VTSS\_JR1\_PACKET\_HDR\_SIZE\_BYTES, 763  
 VTSS\_JR1\_RX\_IFH\_SIZE, 765  
 VTSS\_JR2\_PACKET\_HDR\_SIZE\_BYTES, 764  
 VTSS\_JR2\_RX\_IFH\_SIZE, 765  
 VTSS\_L26\_PACKET\_HDR\_SIZE\_BYTES, 764  
 VTSS\_L26\_RX\_IFH\_SIZE, 765  
 VTSS\_PACKET\_HDR\_SIZE\_BYTES, 764  
 VTSS\_PACKET\_TX\_IFH\_MAX, 765  
 VTSS\_PRIO\_SUPER, 763  
 VTSS\_SVL\_PACKET\_HDR\_SIZE\_BYTES, 764  
 VTSS\_SVL\_RX\_IFH\_SIZE, 764  
 VTSS\_VSTAX\_HDR\_SIZE, 763  
 VTSS\_VSTAX\_TTL\_PORT, 763  
 vtss\_npi\_conf\_get, 769  
 vtss\_npi\_conf\_set, 769  
 vtss\_packet\_dma\_conf\_get, 781  
 vtss\_packet\_dma\_conf\_set, 781  
 vtss\_packet\_dma\_offset, 781  
 vtss\_packet\_filter\_t, 765  
 vtss\_packet\_frame\_filter, 775  
 vtss\_packet\_frame\_info\_init, 775  
 vtss\_packet\_oam\_type\_t, 766  
 vtss\_packet\_port\_filter\_get, 776  
 vtss\_packet\_port\_info\_init, 776  
 vtss\_packet\_ptp\_action\_t, 767  
 vtss\_packet\_rx\_conf\_get, 770  
 vtss\_packet\_rx\_conf\_set, 770  
 vtss\_packet\_rx\_frame\_discard, 773  
 vtss\_packet\_rx\_frame\_get, 772

vtss\_packet\_rx\_frame\_get\_raw, 773  
vtss\_packet\_rx\_hdr\_decode, 778  
vtss\_packet\_rx\_hints\_t, 767  
vtss\_packet\_rx\_port\_conf\_get, 770  
vtss\_packet\_rx\_port\_conf\_set, 772  
vtss\_packet\_tx\_frame, 780  
vtss\_packet\_tx\_frame\_port, 774  
vtss\_packet\_tx\_frame\_port\_vlan, 774  
vtss\_packet\_tx\_frame\_vlan, 775  
vtss\_packet\_tx\_frame\_vstax, 777  
vtss\_packet\_tx\_hdr\_compile, 779  
vtss\_packet\_tx\_hdr\_encode, 778  
vtss\_packet\_tx\_info\_init, 780  
vtss\_packet\_tx\_vstax\_t, 768  
vtss\_packet\_vstax\_frame2header, 777  
vtss\_packet\_vstax\_header2frame, 777  
vtss\_tag\_type\_t, 767  
vtss\_vstax\_fwd\_mode\_t, 766  
vtss\_packet\_dma\_conf\_get  
    vtss\_packet\_api.h, 781  
vtss\_packet\_dma\_conf\_set  
    vtss\_packet\_api.h, 781  
vtss\_packet\_dma\_conf\_t, 165  
    dma\_enable, 165  
vtss\_packet\_dma\_offset  
    vtss\_packet\_api.h, 781  
vtss\_packet\_filter\_t  
    vtss\_packet\_api.h, 765  
vtss\_packet\_frame\_filter  
    vtss\_packet\_api.h, 775  
vtss\_packet\_frame\_info\_init  
    vtss\_packet\_api.h, 775  
vtss\_packet\_frame\_info\_t, 166  
    aggr\_rx\_disable, 167  
    aggr\_tx\_disable, 167  
    glag\_no, 166  
    port\_no, 166  
    port\_tx, 167  
    vid, 166  
vtss\_packet\_oam\_type\_t  
    vtss\_packet\_api.h, 766  
vtss\_packet\_port\_filter\_get  
    vtss\_packet\_api.h, 776  
vtss\_packet\_port\_filter\_t, 167  
    filter, 168  
    tpid, 168  
vtss\_packet\_port\_info\_init  
    vtss\_packet\_api.h, 776  
vtss\_packet\_port\_info\_t, 168  
    aggr\_rx\_disable, 169  
    aggr\_tx\_disable, 170  
    glag\_no, 169  
    port\_no, 169  
    vid, 169  
vtss\_packet\_ptp\_action\_t  
    vtss\_packet\_api.h, 767  
vtss\_packet\_reg\_type\_t  
    types.h, 602  
vtss\_packet\_rx\_conf\_get  
    vtss\_packet\_api.h, 770  
vtss\_packet\_rx\_conf\_set  
    vtss\_packet\_api.h, 770  
vtss\_packet\_rx\_conf\_t, 170  
    grp\_map, 171  
    map, 171  
    queue, 170  
    reg, 171  
vtss\_packet\_rx\_frame\_discard  
    vtss\_packet\_api.h, 773  
vtss\_packet\_rx\_frame\_get  
    vtss\_packet\_api.h, 772  
vtss\_packet\_rx\_frame\_get\_raw  
    vtss\_packet\_api.h, 773  
vtss\_packet\_rx\_grp\_t  
    types.h, 597  
vtss\_packet\_rx\_hdr\_decode  
    vtss\_packet\_api.h, 778  
vtss\_packet\_rx\_header\_t, 171  
    arrived\_tagged, 173  
    learn, 172  
    length, 172  
    port\_no, 172  
    queue\_mask, 172  
    tag, 173  
    vstax, 173  
vtss\_packet\_rx\_hints\_t  
    vtss\_packet\_api.h, 767  
vtss\_packet\_rx\_info\_t, 173  
    acl\_hit, 178  
    acl\_idx, 178  
    cos, 178  
    glag\_no, 175  
    hints, 174  
    hw\_tstamp, 180  
    hw\_tstamp\_decoded, 180  
    isdx, 182  
    length, 175  
    oam\_info, 181  
    oam\_info\_decoded, 182  
    port\_no, 175  
    sflow\_port\_no, 181  
    sflow\_type, 180  
    stripped\_tag, 177  
    sw\_tstamp, 179  
    tag, 176  
    tag\_type, 176  
    tstamp\_id, 179  
    tstamp\_id\_decoded, 179  
    vstax, 182  
    xtr\_qu\_mask, 177  
vtss\_packet\_rx\_meta\_t, 183  
    chip\_no, 184  
    etype, 185  
    fcs, 185  
    length, 186  
    no\_wait, 184

sw\_tstamp, 186  
 xtr\_qu, 184  
 vtss\_packet\_rx\_port\_conf\_get  
     vtss\_packet\_api.h, 770  
 vtss\_packet\_rx\_port\_conf\_set  
     vtss\_packet\_api.h, 772  
 vtss\_packet\_rx\_port\_conf\_t, 187  
     bpdu\_reg, 188  
     garp\_reg, 188  
 vtss\_packet\_rx\_queue\_conf\_t, 188  
     npi, 189  
     size, 188  
 vtss\_packet\_rx\_queue\_map\_t, 189  
     bpdu\_queue, 190  
     garp\_queue, 190  
     igmp\_queue, 190  
     ipmc\_ctrl\_queue, 190  
     learn\_queue, 190  
     lrn\_all\_queue, 191  
     mac\_vid\_queue, 190  
     sflow\_queue, 191  
     stack\_queue, 191  
 vtss\_packet\_rx\_queue\_npi\_conf\_t, 191  
     enable, 192  
 vtss\_packet\_rx\_reg\_t, 192  
     bpdu\_cpu\_only, 193  
     garp\_cpu\_only, 193  
     igmp\_cpu\_only, 193  
     ipmc\_ctrl\_cpu\_copy, 193  
     mld\_cpu\_only, 193  
 vtss\_packet\_tx\_frame  
     vtss\_packet\_api.h, 780  
 vtss\_packet\_tx\_frame\_port  
     vtss\_packet\_api.h, 774  
 vtss\_packet\_tx\_frame\_port\_vlan  
     vtss\_packet\_api.h, 774  
 vtss\_packet\_tx\_frame\_vlan  
     vtss\_packet\_api.h, 775  
 vtss\_packet\_tx\_frame\_vstax  
     vtss\_packet\_api.h, 777  
 vtss\_packet\_tx\_grp\_t  
     types.h, 597  
 vtss\_packet\_tx\_hdr\_compile  
     vtss\_packet\_api.h, 779  
 vtss\_packet\_tx\_hdr\_encode  
     vtss\_packet\_api.h, 778  
 vtss\_packet\_tx\_ifh\_t, 194  
     ifh, 194  
     length, 194  
 vtss\_packet\_tx\_info\_init  
     vtss\_packet\_api.h, 780  
 vtss\_packet\_tx\_info\_t, 195  
     aggr\_code, 198  
     bin, 200  
     cos, 198  
     dp, 204  
     dst\_port\_mask, 196  
     frm\_len, 197  
     isdx, 203  
     isdx\_dont\_use, 204  
     latch\_timestamp, 202  
     masquerade\_port, 205  
     oam\_type, 203  
     pdu\_offset, 205  
     ptp\_action, 201  
     ptp\_id, 201  
     ptp\_timestamp, 202  
     switch\_frm, 196  
     sym, 200  
     tag, 197  
     tx\_vstax\_hdr, 199  
     vstax, 201  
 vtss\_packet\_tx\_vstax\_t  
     vtss\_packet\_api.h, 768  
 vtss\_packet\_vstax\_frame2header  
     vtss\_packet\_api.h, 777  
 vtss\_packet\_vstax\_header2frame  
     vtss\_packet\_api.h, 777  
 vtss\_phy\_10G\_is\_valid  
     vtss\_phy\_10g\_api.h, 856  
 vtss\_phy\_10g\_apc\_conf\_get  
     vtss\_phy\_10g\_api.h, 836  
 vtss\_phy\_10g\_apc\_conf\_set  
     vtss\_phy\_10g\_api.h, 836  
 vtss\_phy\_10g\_apc\_conf\_t, 206  
     op\_mode, 206  
     op\_mode\_flag, 206  
 vtss\_phy\_10g\_apc\_restart  
     vtss\_phy\_10g\_api.h, 837  
 vtss\_phy\_10g\_apc\_status\_get  
     vtss\_phy\_10g\_api.h, 837  
 vtss\_phy\_10g\_apc\_status\_t, 207  
     freeze, 207  
     reset, 207  
 vtss\_phy\_10g\_api.h  
     AMPLITUDE\_POINTS, 798  
     apc\_ib\_regulator\_t, 815  
     BOOLEAN\_STORAGE\_COUNT, 798  
     ckout\_sel\_, 822  
     ckout\_sel\_t, 811  
     clk\_mstr\_t, 816  
     ddr\_mode\_t, 815  
     oper\_mode\_t, 812  
     PHASE\_POINTS, 798  
     UNSIGNED\_STORAGE\_COUNT, 798  
     VTSS\_10G\_PHY\_GPIO\_MAL\_MAX, 799  
     VTSS\_10G\_PHY\_GPIO\_MAX, 799  
     VTSS\_PHY\_10G\_FEC\_FIXED\_CNT\_THRESH↔\_EV, 808  
     VTSS\_PHY\_10G\_FEC\_UNFIXED\_CNT\_THRE↔\_SH\_EV, 808  
     VTSS\_PHY\_10G\_GPIO\_INT\_AGG0\_EV, 809  
     VTSS\_PHY\_10G\_GPIO\_INT\_AGG1\_EV, 809  
     VTSS\_PHY\_10G\_GPIO\_INT\_AGG2\_EV, 809  
     VTSS\_PHY\_10G\_GPIO\_INT\_AGG3\_EV, 809  
     VTSS\_PHY\_10G\_HIGH\_BER\_EV, 800

VTSS\_PHY\_10G\_HIGHBER\_EV, 808  
VTSS\_PHY\_10G\_HOST\_MAC\_LOCAL\_FAULT←\_EV, 810  
VTSS\_PHY\_10G\_HOST\_MAC\_REMOTE\_FAU←\_LT\_EV, 811  
VTSS\_PHY\_10G\_LINE\_MAC\_LOCAL\_FAULT←\_EV, 810  
VTSS\_PHY\_10G\_LINE\_MAC\_REMOTE\_FAU←\_T\_EV, 810  
VTSS\_PHY\_10G\_LINK\_LOS\_EV, 800  
VTSS\_PHY\_10G\_LOPC\_EV, 800  
VTSS\_PHY\_10G\_MACSEC\_DISABLED, 799  
VTSS\_PHY\_10G\_MACSEC\_KEY\_128, 799  
VTSS\_PHY\_10G\_MODULE\_STAT\_EV, 801  
VTSS\_PHY\_10G\_ONE\_LINE\_ACTIVE, 798  
VTSS\_PHY\_10G\_PCS\_RECEIVE\_FAULT\_EV, 801  
VTSS\_PHY\_10G\_RX\_BLK\_DEC\_CNT\_THRES←\_H\_EV, 807  
VTSS\_PHY\_10G\_RX\_CHAR\_DEC\_CNT\_THR←\_ESH\_EV, 807  
VTSS\_PHY\_10G\_RX\_LINK\_STAT\_EV, 808, 809  
VTSS\_PHY\_10G\_RX\_LOL\_EV, 800, 806  
VTSS\_PHY\_10G\_RX\_LOS\_EV, 806  
VTSS\_PHY\_10G\_RX\_SEQ\_CNT\_THRESH\_EV, 807  
VTSS\_PHY\_10G\_TIMESTAMP\_DISABLED, 799  
VTSS\_PHY\_10G\_TX\_BLK\_ENC\_CNT\_THRES←\_H\_EV, 807  
VTSS\_PHY\_10G\_TX\_CHAR\_ENC\_CNT\_THRE←\_SH\_EV, 807  
VTSS\_PHY\_10G\_TX\_LOL\_EV, 800, 806  
VTSS\_PHY\_10G\_TX\_SEQ\_CNT\_THRESH\_EV, 808  
VTSS\_PHY\_1G\_HOST\_AUTONEG\_RESTART←\_EV, 810  
VTSS\_PHY\_1G\_LINE\_AUTONEG\_RESTART←\_EV, 810  
VTSS\_PHY\_EWIS\_AISL\_EV, 802  
VTSS\_PHY\_EWIS\_AISP\_EV, 803  
VTSS\_PHY\_EWIS\_B1\_NZ\_EV, 804  
VTSS\_PHY\_EWIS\_B1\_THRESH\_EV, 805  
VTSS\_PHY\_EWIS\_B2\_NZ\_EV, 804  
VTSS\_PHY\_EWIS\_B2\_THRESH\_EV, 805  
VTSS\_PHY\_EWIS\_B3\_NZ\_EV, 804  
VTSS\_PHY\_EWIS\_B3\_THRESH\_EV, 805  
VTSS\_PHY\_EWIS\_FAIS\_EV, 801  
VTSS\_PHY\_EWIS\_FERDIP\_EV, 803  
VTSS\_PHY\_EWIS\_FEUNEQP\_EV, 803  
VTSS\_PHY\_EWIS\_FPLM\_EV, 801  
VTSS\_PHY\_EWIS\_LCDP\_EV, 802  
VTSS\_PHY\_EWIS\_LOF\_EV, 802  
VTSS\_PHY\_EWIS\_LOPP\_EV, 803  
VTSS\_PHY\_EWIS\_PLMP\_EV, 802  
VTSS\_PHY\_EWIS\_RDIL\_EV, 802  
VTSS\_PHY\_EWIS\_REIL\_EV, 804  
VTSS\_PHY\_EWIS\_REIL\_NZ\_EV, 805  
VTSS\_PHY\_EWIS\_REIL\_THRESH\_EV, 806  
VTSS\_PHY\_EWIS\_REIP\_EV, 804  
VTSS\_PHY\_EWIS\_REIP\_NZ\_EV, 805  
VTSS\_PHY\_EWIS\_REIP\_THRESH\_EV, 806  
VTSS\_PHY\_EWIS\_SEF\_EV, 801  
VTSS\_PHY\_EWIS\_UNEQP\_EV, 803  
VTSS\_SLEWRATE\_120PS, 797  
VTSS\_SLEWRATE\_25PS, 796  
VTSS\_SLEWRATE\_35PS, 797  
VTSS\_SLEWRATE\_55PS, 797  
VTSS\_SLEWRATE\_70PS, 797  
VTSS\_SLEWRATE\_INVALID, 797  
vtss\_10g\_phy\_gpio\_t, 827  
vtss\_32\_cntr\_t, 811  
vtss\_channel\_t, 818  
vtss\_ckout\_data\_sel\_t, 819  
vtss\_gpio\_10g\_aggr\_intrpt\_t, 831  
vtss\_gpio\_10g\_chan\_intrpt\_t, 830  
vtss\_gpio\_10g\_gpio\_intr\_sgnl\_t, 828  
vtss\_gpio\_10g\_input\_t, 831  
vtss\_gpio\_10g\_no\_t, 811  
vtss\_lb\_type\_t, 824  
vtss\_phy\_10G\_is\_valid, 856  
vtss\_phy\_10g\_apc\_conf\_get, 836  
vtss\_phy\_10g\_apc\_conf\_set, 836  
vtss\_phy\_10g\_apc\_restart, 837  
vtss\_phy\_10g\_apc\_status\_get, 837  
vtss\_phy\_10g\_auto\_failover\_event\_t, 825  
vtss\_phy\_10g\_auto\_failover\_filter\_t, 826  
vtss\_phy\_10g\_auto\_failover\_get, 858  
vtss\_phy\_10g\_auto\_failover\_set, 858  
vtss\_phy\_10g\_base\_host\_kr\_train\_aneg\_set, 848  
vtss\_phy\_10g\_base\_kr\_conf\_get, 849  
vtss\_phy\_10g\_base\_kr\_conf\_set, 849  
vtss\_phy\_10g\_base\_kr\_host\_conf\_get, 850  
vtss\_phy\_10g\_base\_kr\_host\_conf\_set, 850  
vtss\_phy\_10g\_base\_kr\_train\_aneg\_get, 847  
vtss\_phy\_10g\_base\_kr\_train\_aneg\_set, 848  
vtss\_phy\_10g\_ckout\_conf\_set, 844  
vtss\_phy\_10g\_ckout\_freq\_t, 819  
vtss\_phy\_10g\_clause\_37\_control\_get, 853  
vtss\_phy\_10g\_clause\_37\_control\_set, 854  
vtss\_phy\_10g\_clause\_37\_remote\_fault\_t, 823  
vtss\_phy\_10g\_clause\_37\_status\_get, 853  
vtss\_phy\_10g\_clk\_sel\_t, 821  
vtss\_phy\_10g\_cnt\_get, 855  
vtss\_phy\_10g\_csr\_read, 873  
vtss\_phy\_10g\_csr\_write, 873  
vtss\_phy\_10g\_debug\_register\_dump, 847  
vtss\_phy\_10g\_edc\_fw\_status\_get, 872  
vtss\_phy\_10g\_event\_enable\_get, 869  
vtss\_phy\_10g\_event\_enable\_set, 868  
vtss\_phy\_10g\_event\_poll, 870  
vtss\_phy\_10g\_event\_t, 811  
vtss\_phy\_10g\_extended\_event\_enable\_get, 869  
vtss\_phy\_10g\_extended\_event\_enable\_set, 871  
vtss\_phy\_10g\_extended\_event\_poll, 871  
vtss\_phy\_10g\_extnd\_event\_t, 812  
vtss\_phy\_10g\_failover\_get, 857

vtss\_phy\_10g\_failover\_mode\_t, 825  
 vtss\_phy\_10g\_failover\_set, 857  
 vtss\_phy\_10g\_fc\_buffer\_reset, 872  
 vtss\_phy\_10g\_get\_user\_data, 875  
 vtss\_phy\_10g\_gpio\_mode\_get, 867  
 vtss\_phy\_10g\_gpio\_mode\_set, 867  
 vtss\_phy\_10g\_gpio\_read, 868  
 vtss\_phy\_10g\_gpio\_write, 868  
 vtss\_phy\_10g\_host\_clk\_conf\_set, 845  
 vtss\_phy\_10g\_host\_recvrd\_clk\_conf\_set, 846  
 vtss\_phy\_10g\_i2c\_read, 874  
 vtss\_phy\_10g\_i2c\_write, 875  
 vtss\_phy\_10g\_ib\_apc\_op\_mode\_t, 817  
 vtss\_phy\_10g\_ib\_conf\_get, 835  
 vtss\_phy\_10g\_ib\_conf\_set, 834  
 vtss\_phy\_10g\_ib\_status\_get, 835  
 vtss\_phy\_10g\_id\_get, 866  
 vtss\_phy\_10g\_init, 832  
 vtss\_phy\_10g\_jitter\_conf\_get, 838  
 vtss\_phy\_10g\_jitter\_conf\_set, 838  
 vtss\_phy\_10g\_jitter\_status\_get, 839  
 vtss\_phy\_10g\_kr\_status\_get, 851  
 vtss\_phy\_10g\_lane\_sync\_set, 846  
 vtss\_phy\_10g\_line\_clk\_conf\_set, 845  
 vtss\_phy\_10g\_line\_recvrd\_clk\_conf\_set, 846  
 vtss\_phy\_10g\_loopback\_get, 855  
 vtss\_phy\_10g\_loopback\_set, 854  
 vtss\_phy\_10g\_media\_t, 816  
 vtss\_phy\_10g\_mode\_get, 832  
 vtss\_phy\_10g\_mode\_set, 834  
 vtss\_phy\_10g\_ob\_status\_get, 851  
 vtss\_phy\_10g\_pcs\_prbs\_gen\_conf\_get, 860  
 vtss\_phy\_10g\_pcs\_prbs\_gen\_conf\_set, 860  
 vtss\_phy\_10g\_pcs\_prbs\_mon\_conf\_get, 861  
 vtss\_phy\_10g\_pcs\_prbs\_mon\_conf\_set, 861  
 vtss\_phy\_10g\_pcs\_prbs\_mon\_status\_get, 862  
 vtss\_phy\_10g\_pcs\_status\_get, 870  
 vtss\_phy\_10g\_pkt\_gen\_conf, 865  
 vtss\_phy\_10g\_pkt\_mon\_conf, 865  
 vtss\_phy\_10g\_pkt\_mon\_counters\_get, 866  
 vtss\_phy\_10g\_pkt\_mon\_rst\_t, 826  
 vtss\_phy\_10g\_poll\_1sec, 871  
 vtss\_phy\_10g\_power\_get, 856  
 vtss\_phy\_10g\_power\_set, 856  
 vtss\_phy\_10g\_power\_t, 825  
 vtss\_phy\_10g\_prbs\_gen\_conf, 862  
 vtss\_phy\_10g\_prbs\_gen\_conf\_get, 863  
 vtss\_phy\_10g\_prbs\_mon\_conf, 863  
 vtss\_phy\_10g\_prbs\_mon\_conf\_get, 864  
 vtss\_phy\_10g\_prbs\_mon\_status\_get, 864  
 vtss\_phy\_10g\_recvrd\_clk\_sel\_t, 822  
 vtss\_phy\_10g\_reset, 852  
 vtss\_phy\_10g\_rx\_macro\_t, 823  
 vtss\_phy\_10g\_rxckout\_get, 841  
 vtss\_phy\_10g\_rxckout\_set, 841  
 vtss\_phy\_10g\_sckout\_conf\_set, 844  
 vtss\_phy\_10g\_sckout\_freq\_t, 822  
 vtss\_phy\_10g\_serdes\_status\_get, 852  
 vtss\_phy\_10g\_sgmii\_mode\_set, 874  
 vtss\_phy\_10g\_squelch\_src\_t, 820  
 vtss\_phy\_10g\_srefclk\_conf\_get, 843  
 vtss\_phy\_10g\_srefclk\_conf\_set, 843  
 vtss\_phy\_10g\_srefclk\_freq\_t, 818  
 vtss\_phy\_10g\_status\_get, 852  
 vtss\_phy\_10g\_sync\_e\_clkout\_get, 839  
 vtss\_phy\_10g\_sync\_e\_clkout\_set, 840  
 vtss\_phy\_10g\_tx\_macro\_t, 823  
 vtss\_phy\_10g\_txckout\_get, 842  
 vtss\_phy\_10g\_txckout\_set, 842  
 vtss\_phy\_10g\_vscope\_conf\_get, 859  
 vtss\_phy\_10g\_vscope\_conf\_set, 859  
 vtss\_phy\_10g\_vscope\_scan\_status\_get, 859  
 vtss\_phy\_10g\_vscope\_scan\_t, 826  
 vtss\_phy\_10g\_xfp\_clkout\_get, 840  
 vtss\_phy\_10g\_xfp\_clkout\_set, 841  
 vtss\_phy\_6g\_link\_partner\_distance\_t, 817  
 vtss\_phy\_interface\_mode, 813  
 vtss\_phy\_warm\_start\_10g\_failed\_get, 874  
 vtss\_recvrd\_clkout\_t, 818  
 vtss\_recvrd\_t, 814  
 vtss\_recvrdclk\_cdr\_div\_t, 814  
 vtss\_rptr\_rate\_t, 816  
 vtss\_srefclk\_div\_t, 814  
 vtss\_wref\_clk\_div\_t, 815  
 vtss\_wrefclk\_t, 813  
 vtss\_phy\_10g\_auto\_failover\_conf\_t, 208  
 a\_gpio, 209  
 channel\_id, 209  
 enable, 210  
 evnt, 208  
 filter, 210  
 fltr\_val, 210  
 is\_host\_side, 209  
 port\_no, 208  
 trig\_ch\_id, 209  
 v\_gpio, 209  
 vtss\_phy\_10g\_auto\_failover\_event\_t  
     vtss\_phy\_10g\_api.h, 825  
 vtss\_phy\_10g\_auto\_failover\_filter\_t  
     vtss\_phy\_10g\_api.h, 826  
 vtss\_phy\_10g\_auto\_failover\_get  
     vtss\_phy\_10g\_api.h, 858  
 vtss\_phy\_10g\_auto\_failover\_set  
     vtss\_phy\_10g\_api.h, 858  
 vtss\_phy\_10g\_base\_host\_kr\_train\_aneg\_set  
     vtss\_phy\_10g\_api.h, 848  
 vtss\_phy\_10g\_base\_kr\_autoneg\_t, 210  
     an\_enable, 211  
     an\_reset, 211  
     an\_restart, 211  
 vtss\_phy\_10g\_base\_kr\_conf\_get  
     vtss\_phy\_10g\_api.h, 849  
 vtss\_phy\_10g\_base\_kr\_conf\_set  
     vtss\_phy\_10g\_api.h, 849  
 vtss\_phy\_10g\_base\_kr\_conf\_t, 212  
     ampl, 213

c0, 212  
c1, 212  
cm1, 212  
en\_ob, 213  
ser\_inv, 213  
slewrate, 213  
vtss\_phy\_10g\_base\_kr\_host\_conf\_get  
    vtss\_phy\_10g\_api.h, 850  
vtss\_phy\_10g\_base\_kr\_host\_conf\_set  
    vtss\_phy\_10g\_api.h, 850  
vtss\_phy\_10g\_base\_kr\_id\_adv\_abil\_t, 214  
    adv\_10g, 214  
    adv\_1g, 214  
    fec\_abil, 214  
    fec\_req, 215  
vtss\_phy\_10g\_base\_kr\_status\_t, 215  
    aneg, 215  
    fec, 216  
    train, 216  
vtss\_phy\_10g\_base\_kr\_train\_aneg\_get  
    vtss\_phy\_10g\_api.h, 847  
vtss\_phy\_10g\_base\_kr\_train\_aneg\_set  
    vtss\_phy\_10g\_api.h, 848  
vtss\_phy\_10g\_base\_kr\_train\_aneg\_t, 216  
    autoneg, 217  
    host\_kr, 217  
    id\_abil, 217  
    line\_kr, 217  
    training, 217  
vtss\_phy\_10g\_base\_kr\_training\_t, 218  
    enable, 218  
    ld\_pre\_init, 219  
    trmthd\_c0, 219  
    trmthd\_cm, 219  
    trmthd\_cp, 218  
vtss\_phy\_10g\_ckout\_conf\_set  
    vtss\_phy\_10g\_api.h, 844  
vtss\_phy\_10g\_ckout\_conf\_t, 219  
    ckout\_sel, 221  
    enable, 221  
    freq, 220  
    mode, 220  
    squelch\_inv, 220  
    src, 220  
vtss\_phy\_10g\_ckout\_freq\_t  
    vtss\_phy\_10g\_api.h, 819  
vtss\_phy\_10g\_clause\_37\_adv\_t, 221  
    acknowledge, 223  
    asymmetric\_pause, 222  
    fdx, 222  
    hdx, 222  
    next\_page, 223  
    remote\_fault, 222  
    symmetric\_pause, 222  
vtss\_phy\_10g\_clause\_37\_cmn\_status\_t, 223  
    host, 224  
    line, 224  
vtss\_phy\_10g\_clause\_37\_control\_get  
    vtss\_phy\_10g\_api.h, 853  
    vtss\_phy\_10g\_clause\_37\_control\_set  
        vtss\_phy\_10g\_api.h, 854  
    vtss\_phy\_10g\_clause\_37\_control\_t, 224  
        advertisement, 225  
        enable, 225  
        enable\_pass\_thru, 225  
        host, 225  
        l\_h, 225  
        line, 225  
    vtss\_phy\_10g\_clause\_37\_remote\_fault\_t  
        vtss\_phy\_10g\_api.h, 823  
    vtss\_phy\_10g\_clause\_37\_status\_get  
        vtss\_phy\_10g\_api.h, 853  
    vtss\_phy\_10g\_clause\_37\_status\_t, 226  
        autoneg, 227  
        complete, 226  
        link, 226  
        partner\_advertisement, 227  
vtss\_phy\_10g\_clk\_sel\_t  
    vtss\_phy\_10g\_api.h, 821  
vtss\_phy\_10g\_clk\_src\_t, 227  
    is\_high\_amp, 228  
vtss\_phy\_10g\_cnt\_get  
    vtss\_phy\_10g\_api.h, 855  
vtss\_phy\_10g\_cnt\_t, 228  
    pcs, 228  
vtss\_phy\_10g\_csr\_read  
    vtss\_phy\_10g\_api.h, 873  
vtss\_phy\_10g\_csr\_write  
    vtss\_phy\_10g\_api.h, 873  
vtss\_phy\_10g\_debug\_register\_dump  
    vtss\_phy\_10g\_api.h, 847  
vtss\_phy\_10g\_edc\_fw\_status\_get  
    vtss\_phy\_10g\_api.h, 872  
vtss\_phy\_10g\_event\_enable\_get  
    vtss\_phy\_10g\_api.h, 869  
vtss\_phy\_10g\_event\_enable\_set  
    vtss\_phy\_10g\_api.h, 868  
vtss\_phy\_10g\_event\_poll  
    vtss\_phy\_10g\_api.h, 870  
vtss\_phy\_10g\_event\_t  
    vtss\_phy\_10g\_api.h, 811  
vtss\_phy\_10g\_extended\_event\_enable\_get  
    vtss\_phy\_10g\_api.h, 869  
vtss\_phy\_10g\_extended\_event\_enable\_set  
    vtss\_phy\_10g\_api.h, 871  
vtss\_phy\_10g\_extended\_event\_poll  
    vtss\_phy\_10g\_api.h, 871  
vtss\_phy\_10g\_extnd\_event\_t  
    vtss\_phy\_10g\_api.h, 812  
vtss\_phy\_10g\_failover\_get  
    vtss\_phy\_10g\_api.h, 857  
vtss\_phy\_10g\_failover\_mode\_t  
    vtss\_phy\_10g\_api.h, 825  
vtss\_phy\_10g\_failover\_set  
    vtss\_phy\_10g\_api.h, 857  
vtss\_phy\_10g\_fc\_buffer\_reset

vtss\_phy\_10g\_api.h, 872  
 vtss\_phy\_10g\_fifo\_sync\_t, 229  
     bypass\_in\_api, 229  
     pr, 229  
     skip\_rev\_check, 229  
 vtss\_phy\_10g\_fw\_status\_t, 230  
     edc\_fw\_api\_load, 231  
     edc\_fw\_chksum, 230  
     edc\_fw\_rev, 230  
     icpu\_activity, 231  
 vtss\_phy\_10g\_get\_user\_data  
     vtss\_phy\_10g\_api.h, 875  
 vtss\_phy\_10g\_gpio\_mode\_get  
     vtss\_phy\_10g\_api.h, 867  
 vtss\_phy\_10g\_gpio\_mode\_set  
     vtss\_phy\_10g\_api.h, 867  
 vtss\_phy\_10g\_gpio\_read  
     vtss\_phy\_10g\_api.h, 868  
 vtss\_phy\_10g\_gpio\_write  
     vtss\_phy\_10g\_api.h, 868  
 vtss\_phy\_10g\_host\_clk\_conf\_set  
     vtss\_phy\_10g\_api.h, 845  
 vtss\_phy\_10g\_host\_clk\_conf\_t, 231  
     clk\_sel\_no, 232  
     mode, 232  
     recvrd\_clk\_sel, 232  
 vtss\_phy\_10g\_host\_recvrd\_clk\_conf\_set  
     vtss\_phy\_10g\_api.h, 846  
 vtss\_phy\_10g\_i2c\_read  
     vtss\_phy\_10g\_api.h, 874  
 vtss\_phy\_10g\_i2c\_write  
     vtss\_phy\_10g\_api.h, 875  
 vtss\_phy\_10g\_ib\_apc\_op\_mode\_t  
     vtss\_phy\_10g\_api.h, 817  
 vtss\_phy\_10g\_ib\_conf\_get  
     vtss\_phy\_10g\_api.h, 835  
 vtss\_phy\_10g\_ib\_conf\_set  
     vtss\_phy\_10g\_api.h, 834  
 vtss\_phy\_10g\_ib\_conf\_t, 232  
     agc, 234  
     apc\_bit\_mask, 236  
     c, 234  
     config\_bit\_mask, 236  
     dfe1, 234  
     dfe2, 234  
     dfe3, 235  
     dfe4, 235  
     freeze\_bit\_mask, 236  
     gain, 233  
     gainadj, 233  
     is\_host, 236  
     l, 234  
     ld, 235  
     offs, 233  
     prbs, 235  
     prbs\_inv, 235  
 vtss\_phy\_10g\_ib\_status\_get  
     vtss\_phy\_10g\_api.h, 835  
     vtss\_phy\_10g\_ib\_status\_t, 237  
         bit\_errors, 237  
         ib\_conf, 237  
         sig\_det, 237  
 vtss\_phy\_10g\_ib\_storage\_t, 238  
     ib\_storage, 238  
     ib\_storage\_bool, 238  
 vtss\_phy\_10g\_id\_get  
     vtss\_phy\_10g\_api.h, 866  
 vtss\_phy\_10g\_id\_t, 239  
     channel\_id, 240  
     device\_feature\_status, 240  
     family, 240  
     part\_number, 239  
     phy\_api\_base\_no, 240  
     revision, 239  
     type, 240  
 vtss\_phy\_10g\_init  
     vtss\_phy\_10g\_api.h, 832  
 vtss\_phy\_10g\_init\_parm\_t, 241  
     channel\_conf, 241  
 vtss\_phy\_10g\_jitter\_conf\_get  
     vtss\_phy\_10g\_api.h, 838  
 vtss\_phy\_10g\_jitter\_conf\_set  
     vtss\_phy\_10g\_api.h, 838  
 vtss\_phy\_10g\_jitter\_conf\_t, 242  
     incr\_levn, 242  
     levn, 242  
     vtail, 242  
 vtss\_phy\_10g\_jitter\_status\_get  
     vtss\_phy\_10g\_api.h, 839  
 vtss\_phy\_10g\_kr\_status\_aneg\_t, 243  
     active, 243  
     block\_lock, 245  
     complete, 243  
     fec\_enable, 244  
     lp\_anegable, 245  
     request\_10g, 244  
     request\_1g, 244  
     request\_fec\_change, 244  
     sm, 244  
 vtss\_phy\_10g\_kr\_status\_fec\_t, 245  
     corrected\_block\_cnt, 246  
     enable, 246  
     uncorrected\_block\_cnt, 246  
 vtss\_phy\_10g\_kr\_status\_get  
     vtss\_phy\_10g\_api.h, 851  
 vtss\_phy\_10g\_kr\_status\_train\_t, 246  
     c0\_ob\_tap\_result, 247  
     cm\_ob\_tap\_result, 247  
     complete, 247  
     cp\_ob\_tap\_result, 247  
 vtss\_phy\_10g\_lane\_sync\_conf\_t, 248  
     enable, 248  
     rx\_ch, 249  
     rx\_macro, 248  
     tx\_ch, 249  
     tx\_macro, 248

vtss\_phy\_10g\_lane\_sync\_set  
vtss\_phy\_10g\_api.h, 846  
vtss\_phy\_10g\_line\_clk\_conf\_set  
vtss\_phy\_10g\_api.h, 845  
vtss\_phy\_10g\_line\_clk\_conf\_t, 249  
clk\_sel\_no, 250  
mode, 250  
recvrd\_clk\_sel, 250  
vtss\_phy\_10g\_line\_recvrd\_clk\_conf\_set  
vtss\_phy\_10g\_api.h, 846  
vtss\_phy\_10g\_loopback\_get  
vtss\_phy\_10g\_api.h, 855  
vtss\_phy\_10g\_loopback\_set  
vtss\_phy\_10g\_api.h, 854  
vtss\_phy\_10g\_loopback\_t, 250  
enable, 251  
lb\_type, 251  
vtss\_phy\_10g\_media\_t  
vtss\_phy\_10g\_api.h, 816  
vtss\_phy\_10g\_mode\_get  
vtss\_phy\_10g\_api.h, 832  
vtss\_phy\_10g\_mode\_set  
vtss\_phy\_10g\_api.h, 834  
vtss\_phy\_10g\_mode\_t, 251  
apc\_eqz\_offs\_par\_cfg, 258  
apc\_host\_eqz\_ld\_ctrl, 258  
apc\_host\_ld\_ctrl, 257  
apc\_ib\_regulator, 259  
apc\_line\_eqz\_ld\_ctrl, 258  
apc\_line\_ld\_ctrl, 257  
apc\_offs\_ctrl, 256  
cfg0, 257  
channel\_id, 254  
d\_filter, 257  
ddr\_mode, 260  
dig\_offset\_reg, 256  
edc\_fw\_load, 255  
enable\_pass\_thru, 263  
h\_apc\_conf, 262  
h\_clk\_src, 261  
h\_ib\_conf, 262  
h\_media, 262  
h\_offset\_guard, 259  
high\_input\_gain, 254  
hl\_clk\_synth, 254  
ib\_conf, 256  
ib\_ini\_lp, 257  
ib\_max\_lp, 258  
ib\_min\_lp, 258  
interface, 253  
is\_host\_wan, 261  
is\_init, 263  
l\_apc\_conf, 263  
l\_clk\_src, 261  
l\_ib\_conf, 262  
l\_media, 262  
l\_offset\_guard, 259  
link\_6g\_distance, 261  
lref\_for\_host, 261  
master, 260  
ob\_conf, 256  
oper\_mode, 253  
pma\_txratecontrol, 259  
polarity, 260  
rate, 260  
recvrd\_clk, 255  
recvrd\_clk\_div, 255  
sd6g\_calib\_done, 263  
serdes\_conf, 259  
sref\_clk\_div, 255  
use\_conf, 256  
venice\_rev\_a\_los\_detection\_workaround, 260  
wref\_clk\_div, 255  
wrefclk, 253  
xaui\_lane\_flip, 254  
xfi\_pol\_invert, 254  
vtss\_phy\_10g\_ob\_status\_get  
vtss\_phy\_10g\_api.h, 851  
vtss\_phy\_10g\_ob\_status\_t, 264  
c\_ctrl, 264  
d\_fltr, 265  
is\_host, 266  
levn, 265  
r\_ctrl, 264  
slew, 264  
v3, 265  
v4, 265  
v5, 266  
vp, 265  
vtss\_phy\_10g\_pcs\_prbs\_gen\_conf\_get  
vtss\_phy\_10g\_api.h, 860  
vtss\_phy\_10g\_pcs\_prbs\_gen\_conf\_set  
vtss\_phy\_10g\_api.h, 860  
vtss\_phy\_10g\_pcs\_prbs\_gen\_conf\_t, 266  
prbs\_gen, 267  
vtss\_phy\_10g\_pcs\_prbs\_mon\_conf\_get  
vtss\_phy\_10g\_api.h, 861  
vtss\_phy\_10g\_pcs\_prbs\_mon\_conf\_set  
vtss\_phy\_10g\_api.h, 861  
vtss\_phy\_10g\_pcs\_prbs\_mon\_conf\_t, 267  
error\_counter, 267  
prbs\_mon, 267  
vtss\_phy\_10g\_pcs\_prbs\_mon\_status\_get  
vtss\_phy\_10g\_api.h, 862  
vtss\_phy\_10g\_pcs\_status\_get  
vtss\_phy\_10g\_api.h, 870  
vtss\_phy\_10g\_pkt\_gen\_conf  
vtss\_phy\_10g\_api.h, 865  
vtss\_phy\_10g\_pkt\_gen\_conf\_t, 268  
dmac, 270  
enable, 268  
etype, 269  
frame\_single, 269  
frames, 269  
ingress, 269  
ipg\_len, 270

pkt\_len, 270  
 ptp, 269  
 ptp\_ts\_ns, 271  
 ptp\_ts\_sec, 270  
 smac, 270  
 srate, 271  
 vtss\_phy\_10g\_pkt\_mon\_conf  
     vtss\_phy\_10g\_api.h, 865  
 vtss\_phy\_10g\_pkt\_mon\_conf\_t, 271  
     bad\_crc, 272  
     ber, 273  
     enable, 272  
     frag, 273  
     good\_crc, 272  
     lfault, 273  
     reset, 272  
     update, 272  
 vtss\_phy\_10g\_pkt\_mon\_counters\_get  
     vtss\_phy\_10g\_api.h, 866  
 vtss\_phy\_10g\_pkt\_mon\_rst\_t  
     vtss\_phy\_10g\_api.h, 826  
 vtss\_phy\_10g\_polarity\_inv\_t, 273  
     host\_rx, 274  
     host\_tx, 274  
     line\_rx, 274  
     line\_tx, 274  
 vtss\_phy\_10g\_poll\_1sec  
     vtss\_phy\_10g\_api.h, 871  
 vtss\_phy\_10g\_power\_get  
     vtss\_phy\_10g\_api.h, 856  
 vtss\_phy\_10g\_power\_set  
     vtss\_phy\_10g\_api.h, 856  
 vtss\_phy\_10g\_power\_t  
     vtss\_phy\_10g\_api.h, 825  
 vtss\_phy\_10g\_prbs\_gen\_conf  
     vtss\_phy\_10g\_api.h, 862  
 vtss\_phy\_10g\_prbs\_gen\_conf\_get  
     vtss\_phy\_10g\_api.h, 863  
 vtss\_phy\_10g\_prbs\_gen\_conf\_t, 275  
     enable, 275  
     line, 276  
     prbsn\_tx\_io, 276  
     prbsn\_tx\_iw, 276  
     prbsn\_tx\_sel, 275  
 vtss\_phy\_10g\_prbs\_mon\_conf  
     vtss\_phy\_10g\_api.h, 863  
 vtss\_phy\_10g\_prbs\_mon\_conf\_get  
     vtss\_phy\_10g\_api.h, 864  
 vtss\_phy\_10g\_prbs\_mon\_conf\_t, 276  
     active, 280  
     bist\_mode, 279  
     des\_interface\_width, 278  
     enable, 277  
     error\_states, 278  
     error\_status, 279  
     incomplete, 280  
     instable, 280  
     line, 277  
     main\_status, 279  
     max\_bist\_frames, 277  
     no\_of\_errors, 278  
     no\_sync, 280  
     PRBS\_status, 279  
     prbs\_check\_input\_invert, 278  
     prbsn\_sel, 278  
     stuck\_at\_01, 280  
     stuck\_at\_par, 279  
 vtss\_phy\_10g\_prbs\_mon\_status\_get  
     vtss\_phy\_10g\_api.h, 864  
 vtss\_phy\_10g\_recvrd\_clk\_sel\_t  
     vtss\_phy\_10g\_api.h, 822  
 vtss\_phy\_10g\_reset  
     vtss\_phy\_10g\_api.h, 852  
 vtss\_phy\_10g\_rx\_macro\_t  
     vtss\_phy\_10g\_api.h, 823  
 vtss\_phy\_10g\_rxckout\_conf\_t, 281  
     mode, 281  
     squelch\_on\_lopc, 282  
     squelch\_on\_pcs\_fault, 281  
 vtss\_phy\_10g\_rxckout\_get  
     vtss\_phy\_10g\_api.h, 841  
 vtss\_phy\_10g\_rxckout\_set  
     vtss\_phy\_10g\_api.h, 841  
 vtss\_phy\_10g\_sckout\_conf\_set  
     vtss\_phy\_10g\_api.h, 844  
 vtss\_phy\_10g\_sckout\_conf\_t, 282  
     enable, 283  
     freq, 283  
     mode, 282  
     squelch\_inv, 283  
     src, 283  
 vtss\_phy\_10g\_sckout\_freq\_t  
     vtss\_phy\_10g\_api.h, 822  
 vtss\_phy\_10g\_serdes\_status\_get  
     vtss\_phy\_10g\_api.h, 852  
 vtss\_phy\_10g\_serdes\_status\_t, 284  
     h\_pcs, 290  
     h\_pll5g\_fsm\_lock, 285  
     h\_pll5g\_fsm\_stat, 285  
     h\_pll5g\_gain, 285  
     h\_pll5g\_lock\_status, 285  
     h\_pma, 289  
     h\_rx\_rcpll\_fsm\_status, 287  
     h\_rx\_rcpll\_lock\_status, 286  
     h\_rx\_rcpll\_range, 286  
     h\_rx\_rcpll\_vco\_load, 287  
     h\_tx\_rcpll\_fsm\_status, 288  
     h\_tx\_rcpll\_lock\_status, 288  
     h\_tx\_rcpll\_range, 288  
     h\_tx\_rcpll\_vco\_load, 288  
     l\_pcs, 290  
     l\_pll5g\_fsm\_lock, 286  
     l\_pll5g\_fsm\_stat, 286  
     l\_pll5g\_gain, 286  
     l\_pll5g\_lock\_status, 285  
     l\_pma, 290

`l_rx_rcpll_fsm_status`, 288  
`l_rx_rcpll_lock_status`, 287  
`l_rx_rcpll_range`, 287  
`l_rx_rcpll_vco_load`, 287  
`l_tx_rcpll_fsm_status`, 289  
`l_tx_rcpll_lock_status`, 289  
`l_tx_rcpll_range`, 289  
`l_tx_rcpll_vco_load`, 289  
`rcomp`, 284  
`wis`, 290  
`vtss_phy_10g_sgmii_mode_set`  
    `vtss_phy_10g_api.h`, 874  
`vtss_phy_10g_squelch_src_t`  
    `vtss_phy_10g_api.h`, 820  
`vtss_phy_10g_srefclk_conf_get`  
    `vtss_phy_10g_api.h`, 843  
`vtss_phy_10g_srefclk_conf_set`  
    `vtss_phy_10g_api.h`, 843  
`vtss_phy_10g_srefclk_freq_t`  
    `vtss_phy_10g_api.h`, 818  
`vtss_phy_10g_srefclk_mode_t`, 291  
    enable, 291  
    freq, 291  
`vtss_phy_10g_status_get`  
    `vtss_phy_10g_api.h`, 852  
`vtss_phy_10g_status_t`, 291  
    block\_lock, 294  
    hpcs, 293  
    hpcs\_1g, 293  
    hpma, 292  
    lopc\_stat, 294  
    lpes\_1g, 293  
    pcs, 292  
    pma, 292  
    status, 293  
    wis, 292  
    xs, 293  
`vtss_phy_10g_syncce_clkout_get`  
    `vtss_phy_10g_api.h`, 839  
`vtss_phy_10g_syncce_clkout_set`  
    `vtss_phy_10g_api.h`, 840  
`vtss_phy_10g_timestamp_val_t`, 294  
    timestamp, 295  
`vtss_phy_10g_tx_macro_t`  
    `vtss_phy_10g_api.h`, 823  
`vtss_phy_10g_txckout_conf_t`, 295  
    mode, 295  
`vtss_phy_10g_txckout_get`  
    `vtss_phy_10g_api.h`, 842  
`vtss_phy_10g_txckout_set`  
    `vtss_phy_10g_api.h`, 842  
`vtss_phy_10g_vsphere_conf_get`  
    `vtss_phy_10g_api.h`, 859  
`vtss_phy_10g_vsphere_conf_set`  
    `vtss_phy_10g_api.h`, 859  
`vtss_phy_10g_vsphere_conf_t`, 296  
    enable, 296  
    error\_thres, 296  
    line, 296  
    scan\_type, 296  
`vtss_phy_10g_vsphere_scan_conf_t`, 297  
    ber, 299  
    line, 297  
    x\_count, 298  
    x\_incr, 298  
    x\_start, 297  
    y\_count, 298  
    y\_incr, 298  
    y\_start, 298  
`vtss_phy_10g_vsphere_scan_status_get`  
    `vtss_phy_10g_api.h`, 859  
`vtss_phy_10g_vsphere_scan_status_t`, 299  
    amp\_range, 300  
    error\_free\_x, 300  
    error\_free\_y, 300  
    errors, 300  
    scan\_conf, 299  
`vtss_phy_10g_vsphere_scan_t`  
    `vtss_phy_10g_api.h`, 826  
`vtss_phy_10g_xfp_clkout_get`  
    `vtss_phy_10g_api.h`, 840  
`vtss_phy_10g_xfp_clkout_set`  
    `vtss_phy_10g_api.h`, 841  
`vtss_phy_1588_csr_reg_read`  
    `vtss_phy_ts_api.h`, 1017  
`vtss_phy_1588_csr_reg_write`  
    `vtss_phy_ts_api.h`, 1016  
`vtss_phy_1588_debug_reg_read`  
    `vtss_phy_ts_api.h`, 1020  
`vtss_phy_1g_ts_fifo_sync`  
    `vtss_phy_ts_api.h`, 1020  
`vtss_phy_6g_link_partner_distance_t`  
    `vtss_phy_10g_api.h`, 817  
`vtss_phy_aneg_t`, 301  
    asymmetric\_pause, 302  
    speed\_100m\_fdx, 302  
    speed\_100m\_hdx, 301  
    speed\_10m\_fdx, 301  
    speed\_10m\_hdx, 301  
    speed\_1g\_fdx, 302  
    speed\_1g\_hdx, 302  
    symmetric\_pause, 302  
    tx\_remote\_fault, 303  
`vtss_phy_api.h`  
    lb\_type, 905  
    MAX\_CFG\_BUF\_SIZE, 886  
    MAX\_STAT\_BUF\_SIZE, 886  
    MAX\_WOL\_MAC\_ADDR\_SIZE, 897  
    MAX\_WOL\_PASSWD\_SIZE, 897  
    MIN\_WOL\_PASSWD\_SIZE, 897  
    rgmii\_skew\_delay\_psec\_t, 900  
    VTSS\_PHY\_ACTIPHY\_PWR, 887  
    VTSS\_PHY\_LINK\_AMS\_EV, 891  
    VTSS\_PHY\_LINK\_AUTO\_NEG\_COMPLETE\_EV, 892  
    VTSS\_PHY\_LINK\_AUTO\_NEG\_ERROR\_EV, 892

VTSS\_PHY\_LINK\_DOWN\_PWR, 887  
 VTSS\_PHY\_LINK\_EXT\_EEE\_LINKFAIL\_EV, 895  
 VTSS\_PHY\_LINK\_EXT\_EEE\_WAIT\_RX\_TQ\_EV,  
   895  
 VTSS\_PHY\_LINK\_EXT\_EEE\_WAIT\_TS\_EV, 895  
 VTSS\_PHY\_LINK\_EXT\_EEE\_WAKE\_ERR\_EV,  
   894  
 VTSS\_PHY\_LINK\_EXT\_MACSEC\_EGRESS\_EV,  
   896  
 VTSS\_PHY\_LINK\_EXT\_MACSEC\_FC\_BUFF\_←  
   EV, 896  
 VTSS\_PHY\_LINK\_EXT\_MACSEC\_HOST\_MAC←  
   EV, 895  
 VTSS\_PHY\_LINK\_EXT\_MACSEC\_INGRESS\_←  
   EV, 896  
 VTSS\_PHY\_LINK\_EXT\_MACSEC\_LINE\_MAC←  
   EV, 896  
 VTSS\_PHY\_LINK\_EXT\_MEM\_INT\_RING\_EV,  
   896  
 VTSS\_PHY\_LINK\_EXT\_RR\_SW\_COMPL\_EV,  
   895  
 VTSS\_PHY\_LINK\_EXTENDED\_REG\_INT\_EV,  
   894  
 VTSS\_PHY\_LINK\_FALSE\_CARRIER\_INT\_EV,  
   893  
 VTSS\_PHY\_LINK\_FDX\_STATE\_CHANGE\_EV,  
   892  
 VTSS\_PHY\_LINK\_FFAIL\_EV, 891  
 VTSS\_PHY\_LINK\_INLINE\_POW\_DEV\_DETECT←  
   T\_EV, 892  
 VTSS\_PHY\_LINK\_LINK\_SPEED\_DS\_DETECT←  
   EV, 893  
 VTSS\_PHY\_LINK\_LOS\_EV, 891  
 VTSS\_PHY\_LINK\_MASTER\_SLAVE\_RES\_ER←  
   R\_EV, 894  
 VTSS\_PHY\_LINK\_RX\_ER\_INT\_EV, 894  
 VTSS\_PHY\_LINK\_RX\_FIFO\_OVERFLOW\_INT←  
   EV, 893  
 VTSS\_PHY\_LINK\_SPEED\_STATE\_CHANGE←  
   EV, 892  
 VTSS\_PHY\_LINK\_SYMBOL\_ERR\_INT\_EV, 893  
 VTSS\_PHY\_LINK\_TX\_FIFO\_OVERFLOW\_INT←  
   EV, 893  
 VTSS\_PHY\_LINK\_UP\_FULL\_PWR, 887  
 VTSS\_PHY\_LINK\_WAKE\_ON\_LAN\_INT\_EV, 894  
 VTSS\_PHY\_PAGE\_0x2DAF, 890  
 VTSS\_PHY\_PAGE\_1588, 889  
 VTSS\_PHY\_PAGE\_EXTENDED\_2, 888  
 VTSS\_PHY\_PAGE\_EXTENDED\_3, 888  
 VTSS\_PHY\_PAGE\_EXTENDED\_4, 889  
 VTSS\_PHY\_PAGE\_EXTENDED, 888  
 VTSS\_PHY\_PAGE\_GPIO, 889  
 VTSS\_PHY\_PAGE\_MACSEC, 889  
 VTSS\_PHY\_PAGE\_STANDARD, 888  
 VTSS\_PHY\_PAGE\_TEST, 889  
 VTSS\_PHY\_PAGE\_TR, 890  
 VTSS\_PHY\_POWER\_ACTIPHY\_BIT, 886  
 VTSS\_PHY\_POWER\_DYNAMIC\_BIT, 886  
 VTSS\_PHY\_RECov\_CLK1, 887  
 VTSS\_PHY\_RECov\_CLK2, 887  
 VTSS\_PHY\_RECov\_CLK\_NUM, 888  
 VTSS\_PHY\_REG\_EXTENDED, 890  
 VTSS\_PHY\_REG\_GPIO, 890  
 VTSS\_PHY\_REG\_STANDARD, 890  
 VTSS\_PHY\_REG\_TEST, 891  
 VTSS\_PHY\_REG\_TR, 891  
 vga\_adc\_debug, 925  
 vtss\_fefi\_mode\_t, 905  
 vtss\_phy\_atom12\_patch\_settings\_get, 935  
 vtss\_phy\_cfg\_ib\_cterm, 934  
 vtss\_phy\_cfg\_ob\_post0, 934  
 vtss\_phy\_chip\_temp\_get, 908  
 vtss\_phy\_chip\_temp\_init, 908  
 vtss\_phy\_cl37\_lp\_abil\_get, 911  
 vtss\_phy\_clk\_source\_t, 903  
 vtss\_phy\_clk\_squelch, 904  
 vtss\_phy\_clock\_conf\_get, 915  
 vtss\_phy\_clock\_conf\_set, 914  
 vtss\_phy\_coma\_mode\_disable, 925  
 vtss\_phy\_coma\_mode\_enable, 925  
 vtss\_phy\_conf\_1g\_get, 912  
 vtss\_phy\_conf\_1g\_set, 912  
 vtss\_phy\_conf\_get, 910  
 vtss\_phy\_conf\_set, 910  
 vtss\_phy\_csr\_rd, 931  
 vtss\_phy\_csr\_wr, 930  
 vtss\_phy\_debug\_phyinfo\_print, 937  
 vtss\_phy\_debug\_regdump\_print, 940  
 vtss\_phy\_debug\_register\_dump, 938  
 vtss\_phy\_debug\_stat\_print, 936  
 vtss\_phy\_detect\_base\_ports, 938  
 vtss\_phy\_do\_page\_chk\_get, 932  
 vtss\_phy\_do\_page\_chk\_set, 932  
 vtss\_phy\_eee\_conf\_get, 927  
 vtss\_phy\_eee\_conf\_set, 927  
 vtss\_phy\_eee\_ena, 926  
 vtss\_phy\_eee\_link\_partner\_advertisements\_get,  
   928  
 vtss\_phy\_eee\_power\_save\_state\_get, 927  
 vtss\_phy\_enhanced\_led\_control\_init, 924  
 vtss\_phy\_enhanced\_led\_control\_init\_get, 924  
 vtss\_phy\_event\_enable\_get, 929  
 vtss\_phy\_event\_enable\_set, 928  
 vtss\_phy\_event\_poll, 929  
 vtss\_phy\_ext\_connector\_loopback, 939  
 vtss\_phy\_ext\_event\_poll, 950  
 vtss\_phy\_fast\_link\_fail\_t, 901  
 vtss\_phy\_fefi\_detect, 947  
 vtss\_phy\_fefi\_get, 946  
 vtss\_phy\_fefi\_set, 947  
 vtss\_phy\_flowcontrol\_decode\_status, 936  
 vtss\_phy\_forced\_reset\_t, 900  
 vtss\_phy\_freq\_t, 904  
 vtss\_phy\_gpio\_get, 921  
 vtss\_phy\_gpio\_mode, 920  
 vtss\_phy\_gpio\_mode\_t, 904

vtss\_phy\_gpio\_set, 921  
vtss\_phy\_i2c\_read, 915  
vtss\_phy\_i2c\_write, 916  
vtss\_phy\_id\_get, 911  
vtss\_phy\_is\_8051\_crc\_ok, 933  
vtss\_phy\_is\_viper\_revB, 949  
vtss\_phy\_lcpll\_status\_get, 943  
vtss\_phy\_led\_intensity, 897  
vtss\_phy\_led\_intensity\_get, 923  
vtss\_phy\_led\_intensity\_set, 923  
vtss\_phy\_led\_mode\_set, 922  
vtss\_phy\_led\_mode\_t, 898  
vtss\_phy\_loopback\_get, 933  
vtss\_phy\_loopback\_set, 933  
vtss\_phy\_mac\_media\_inhibit\_odd\_start, 946  
vtss\_phy\_mac\_serid\_pcs\_sgmii\_pre, 902  
vtss\_phy\_macsec\_csr\_sd6g\_rd, 951  
vtss\_phy\_macsec\_csr\_sd6g\_wr, 951  
vtss\_phy\_mdi\_t, 899  
vtss\_phy\_media\_force\_ams\_sel\_t, 903  
vtss\_phy\_media\_interface\_t, 899  
vtss\_phy\_media\_rem\_fault\_t, 902  
vtss\_phy\_mmd\_read, 918  
vtss\_phy\_mmd\_write, 918  
vtss\_phy\_mode\_t, 901  
vtss\_phy\_mse\_1000m\_get, 948  
vtss\_phy\_mse\_100m\_get, 948  
vtss\_phy\_patch\_settings\_get, 942  
vtss\_phy\_pkt\_mode\_t, 900  
vtss\_phy\_port\_eee\_capable, 926  
vtss\_phy\_post\_reset, 906  
vtss\_phy\_power\_conf\_get, 913  
vtss\_phy\_power\_conf\_set, 914  
vtss\_phy\_power\_status\_get, 914  
vtss\_phy\_pre\_reset, 906  
vtss\_phy\_pre\_system\_reset, 907  
vtss\_phy\_read, 917  
vtss\_phy\_read\_page, 917  
vtss\_phy\_read\_tr\_addr, 949  
vtss\_phy\_recov\_clk\_t, 897  
vtss\_phy\_reg\_decode\_status, 935  
vtss\_phy\_reset, 907  
vtss\_phy\_reset\_get, 908  
vtss\_phy\_reset\_lcpll, 944  
vtss\_phy\_sd6g\_ob\_lev\_rd, 945  
vtss\_phy\_sd6g\_ob\_lev\_wr, 946  
vtss\_phy\_sd6g\_ob\_post\_rd, 944  
vtss\_phy\_sd6g\_ob\_post\_wr, 945  
vtss\_phy\_serdes1g\_rcpll\_status\_get, 943  
vtss\_phy\_serdes6g\_rcpll\_status\_get, 942  
vtss\_phy\_serdes\_fmedia\_loopback, 939  
vtss\_phy\_serdes\_sgmii\_loopback, 939  
vtss\_phy\_sigdet\_polarity\_t, 901  
vtss\_phy\_statistic\_get, 931  
vtss\_phy\_status\_1g\_get, 913  
vtss\_phy\_status\_get, 911  
vtss\_phy\_status\_inst\_poll, 950  
vtss\_phy\_unidirectional\_t, 902  
vtss\_phy\_veriphy\_get, 922  
vtss\_phy\_veriphy\_start, 922  
vtss\_phy\_warm\_start\_failed\_get, 937  
vtss\_phy\_wol\_conf\_get, 941  
vtss\_phy\_wol\_conf\_set, 941  
vtss\_phy\_wol\_enable, 940  
vtss\_phy\_write, 919  
vtss\_phy\_write\_masked, 919  
vtss\_phy\_write\_masked\_page, 920  
vtss\_squelch\_workaround, 930  
vtss\_wol\_passwd\_len\_type\_t, 905  
vtss\_phy\_atom12\_patch\_settings\_get  
    vtss\_phy\_api.h, 935  
vtss\_phy\_cfg\_ib\_cterm  
    vtss\_phy\_api.h, 934  
vtss\_phy\_cfg\_ob\_post0  
    vtss\_phy\_api.h, 934  
vtss\_phy\_chip\_temp\_get  
    vtss\_phy\_api.h, 908  
vtss\_phy\_chip\_temp\_init  
    vtss\_phy\_api.h, 908  
vtss\_phy\_cl37\_lp\_abil\_get  
    vtss\_phy\_api.h, 911  
vtss\_phy\_clk\_source\_t  
    vtss\_phy\_api.h, 903  
vtss\_phy\_clk\_squelch  
    vtss\_phy\_api.h, 904  
vtss\_phy\_clock\_conf\_get  
    vtss\_phy\_api.h, 915  
vtss\_phy\_clock\_conf\_set  
    vtss\_phy\_api.h, 914  
vtss\_phy\_clock\_conf\_t, 303  
    freq, 304  
    squelch, 304  
    src, 303  
vtss\_phy\_coma\_mode\_disable  
    vtss\_phy\_api.h, 925  
vtss\_phy\_coma\_mode\_enable  
    vtss\_phy\_api.h, 925  
vtss\_phy\_conf\_1g\_get  
    vtss\_phy\_api.h, 912  
vtss\_phy\_conf\_1g\_set  
    vtss\_phy\_api.h, 912  
vtss\_phy\_conf\_1g\_t, 304  
    cfg, 305  
    master, 305  
    val, 305  
vtss\_phy\_conf\_get  
    vtss\_phy\_api.h, 910  
vtss\_phy\_conf\_set  
    vtss\_phy\_api.h, 910  
vtss\_phy\_conf\_t, 305  
    aneg, 306  
    flf, 307  
    force\_ams\_sel, 308  
    forced, 306  
    mac\_if\_pcs, 307  
    mdi, 306

media\_if\_pcs, 307  
mode, 306  
sigdet, 307  
unidir, 307  
vtss\_phy\_csr\_rd  
    vtss\_phy\_api.h, 931  
vtss\_phy\_csr\_wr  
    vtss\_phy\_api.h, 930  
vtss\_phy\_daisy\_chain\_conf\_t, 308  
    spi\_daisy\_input, 308  
    spi\_daisy\_output, 309  
vtss\_phy\_daisy\_conf\_get  
    vtss\_phy\_ts\_api.h, 997  
vtss\_phy\_daisy\_conf\_set  
    vtss\_phy\_ts\_api.h, 997  
vtss\_phy\_debug\_phyinfo\_print  
    vtss\_phy\_api.h, 937  
vtss\_phy\_debug\_regdump\_print  
    vtss\_phy\_api.h, 940  
vtss\_phy\_debug\_register\_dump  
    vtss\_phy\_api.h, 938  
vtss\_phy\_debug\_stat\_print  
    vtss\_phy\_api.h, 936  
vtss\_phy\_detect\_base\_ports  
    vtss\_phy\_api.h, 938  
vtss\_phy\_do\_page\_chk\_get  
    vtss\_phy\_api.h, 932  
vtss\_phy\_do\_page\_chk\_set  
    vtss\_phy\_api.h, 932  
vtss\_phy\_eee\_conf\_get  
    vtss\_phy\_api.h, 927  
vtss\_phy\_eee\_conf\_set  
    vtss\_phy\_api.h, 927  
vtss\_phy\_eee\_conf\_t, 309  
    eee\_ena\_phy, 310  
    eee\_mode, 309  
vtss\_phy\_eee\_ena  
    vtss\_phy\_api.h, 926  
vtss\_phy\_eee\_link\_partner\_advertisements\_get  
    vtss\_phy\_api.h, 928  
vtss\_phy\_eee\_power\_save\_state\_get  
    vtss\_phy\_api.h, 927  
vtss\_phy\_enhanced\_led\_control\_init  
    vtss\_phy\_api.h, 924  
vtss\_phy\_enhanced\_led\_control\_init\_get  
    vtss\_phy\_api.h, 924  
vtss\_phy\_enhanced\_led\_control\_t, 310  
    ser\_led\_frame\_rate, 311  
    ser\_led\_output\_1, 310  
    ser\_led\_output\_2, 311  
    ser\_led\_select, 311  
vtss\_phy\_event\_enable\_get  
    vtss\_phy\_api.h, 929  
vtss\_phy\_event\_enable\_set  
    vtss\_phy\_api.h, 928  
vtss\_phy\_event\_poll  
    vtss\_phy\_api.h, 929  
vtss\_phy\_ext\_connector\_loopback  
    vtss\_phy\_api.h, 939  
vtss\_phy\_ext\_event\_poll  
    vtss\_phy\_api.h, 950  
vtss\_phy\_fast\_link\_fail\_t  
    vtss\_phy\_api.h, 901  
vtss\_phy\_fefi\_detect  
    vtss\_phy\_api.h, 947  
vtss\_phy\_fefi\_get  
    vtss\_phy\_api.h, 946  
vtss\_phy\_fefi\_set  
    vtss\_phy\_api.h, 947  
vtss\_phy\_flowcontrol\_decode\_status  
    vtss\_phy\_api.h, 936  
vtss\_phy\_forced\_reset\_t  
    vtss\_phy\_api.h, 900  
vtss\_phy\_forced\_t, 311  
    fdx, 312  
    speed, 312  
vtss\_phy\_freq\_t  
    vtss\_phy\_api.h, 904  
vtss\_phy\_gpio\_get  
    vtss\_phy\_api.h, 921  
vtss\_phy\_gpio\_mode  
    vtss\_phy\_api.h, 920  
vtss\_phy\_gpio\_mode\_t  
    vtss\_phy\_api.h, 904  
vtss\_phy\_gpio\_set  
    vtss\_phy\_api.h, 921  
vtss\_phy\_i2c\_read  
    vtss\_phy\_api.h, 915  
vtss\_phy\_i2c\_write  
    vtss\_phy\_api.h, 916  
vtss\_phy\_id\_get  
    vtss\_phy\_api.h, 911  
vtss\_phy\_interface\_mode  
    vtss\_phy\_10g\_api.h, 813  
vtss\_phy\_is\_8051\_crc\_ok  
    vtss\_phy\_api.h, 933  
vtss\_phy\_is\_viper\_revB  
    vtss\_phy\_api.h, 949  
vtss\_phy\_lcpll\_status\_get  
    vtss\_phy\_api.h, 943  
vtss\_phy\_led\_intensity  
    vtss\_phy\_api.h, 897  
vtss\_phy\_led\_intensity\_get  
    vtss\_phy\_api.h, 923  
vtss\_phy\_led\_intensity\_set  
    vtss\_phy\_api.h, 923  
vtss\_phy\_led\_mode\_select\_t, 312  
    mode, 313  
    number, 313  
vtss\_phy\_led\_mode\_set  
    vtss\_phy\_api.h, 922  
vtss\_phy\_led\_mode\_t  
    vtss\_phy\_api.h, 898  
vtss\_phy\_loopback\_get  
    vtss\_phy\_api.h, 933  
vtss\_phy\_loopback\_set

vtss\_phy\_api.h, 933  
vtss\_phy\_loopback\_t, 313  
connector\_enable, 314  
far\_end\_enable, 314  
mac\_serd़es\_equipment\_enable, 315  
mac\_serd़es\_facility\_enable, 314  
mac\_serd़es\_input\_enable, 314  
media\_serd़es\_equipment\_enable, 315  
media\_serd़es\_facility\_enable, 315  
media\_serd़es\_input\_enable, 315  
near\_end\_enable, 314  
vtss\_phy\_ltc\_freq\_synth\_s, 316  
enable, 316  
high\_duty\_cycle, 316  
low\_duty\_cycle, 316  
vtss\_phy\_mac\_media\_inhibit\_odd\_start  
vtss\_phy\_api.h, 946  
vtss\_phy\_mac\_serd\_pcs\_cntl\_t, 317  
aneg\_restart, 318  
disable, 317  
fast\_link\_stat\_ena, 319  
force\_adv\_ability, 318  
inhibit\_odd\_start, 320  
pd\_enable, 318  
restart, 318  
serdes\_aneg\_ena, 319  
serdes\_pol\_inv\_in, 319  
serdes\_pol\_inv\_out, 319  
sgmii\_in\_pre, 318  
sgmii\_out\_pre, 319  
vtss\_phy\_mac\_serd\_pcs\_sgmii\_pre  
vtss\_phy\_api.h, 902  
vtss\_phy\_macsec\_csr\_sd6g\_rd  
vtss\_phy\_api.h, 951  
vtss\_phy\_macsec\_csr\_sd6g\_wr  
vtss\_phy\_api.h, 951  
vtss\_phy\_mdi\_t  
vtss\_phy\_api.h, 899  
vtss\_phy\_media\_force\_ams\_sel\_t  
vtss\_phy\_api.h, 903  
vtss\_phy\_media\_interface\_t  
vtss\_phy\_api.h, 899  
vtss\_phy\_media\_rem\_fault\_t  
vtss\_phy\_api.h, 902  
vtss\_phy\_media\_serd\_pcs\_cntl\_t, 320  
aneg\_pd\_detect, 321  
force\_adv\_ability, 321  
force\_fefi, 322  
force\_fefi\_value, 322  
force\_hls, 322  
inhibit\_odd\_start, 321  
remote\_fault, 321  
serdes\_pol\_inv\_in, 321  
serdes\_pol\_inv\_out, 321  
vtss\_phy\_mmd\_read  
vtss\_phy\_api.h, 918  
vtss\_phy\_mmd\_write  
vtss\_phy\_api.h, 918  
vtss\_phy\_mode\_t  
vtss\_phy\_api.h, 901  
vtss\_phy\_mse\_1000m\_get  
vtss\_phy\_api.h, 948  
vtss\_phy\_mse\_100m\_get  
vtss\_phy\_api.h, 948  
vtss\_phy\_patch\_settings\_get  
vtss\_phy\_api.h, 942  
vtss\_phy\_pcs\_cnt\_t, 322  
ber\_cnt, 323  
block\_lock\_latched, 323  
err\_blk\_cnt, 323  
high\_ber\_latched, 323  
vtss\_phy\_pkt\_mode\_t  
vtss\_phy\_api.h, 900  
vtss\_phy\_port\_eee\_capable  
vtss\_phy\_api.h, 926  
vtss\_phy\_post\_reset  
vtss\_phy\_api.h, 906  
vtss\_phy\_power\_conf\_get  
vtss\_phy\_api.h, 913  
vtss\_phy\_power\_conf\_set  
vtss\_phy\_api.h, 914  
vtss\_phy\_power\_conf\_t, 324  
mode, 324  
vtss\_phy\_power\_mode\_t  
phy.h, 555  
vtss\_phy\_power\_status\_get  
vtss\_phy\_api.h, 914  
vtss\_phy\_power\_status\_t, 325  
level, 325  
vtss\_phy\_pre\_reset  
vtss\_phy\_api.h, 906  
vtss\_phy\_pre\_system\_reset  
vtss\_phy\_api.h, 907  
vtss\_phy\_read  
vtss\_phy\_api.h, 917  
vtss\_phy\_read\_page  
vtss\_phy\_api.h, 917  
vtss\_phy\_read\_tr\_addr  
vtss\_phy\_api.h, 949  
vtss\_phy\_recov\_clk\_t  
vtss\_phy\_api.h, 897  
vtss\_phy\_reg\_decode\_status  
vtss\_phy\_api.h, 935  
vtss\_phy\_reset  
vtss\_phy\_api.h, 907  
vtss\_phy\_reset\_conf\_t, 325  
force, 326  
i\_cpu\_en, 327  
mac\_if, 326  
media\_if, 326  
pkt\_mode, 327  
rgmii, 326  
tbi, 326  
vtss\_phy\_reset\_get  
vtss\_phy\_api.h, 908  
vtss\_phy\_reset\_lcpll

vtss\_phy\_api.h, 944  
 vtss\_phy\_rgmiic\_conf\_t, 327  
   rx\_clk\_skew\_ps, 328  
   tx\_clk\_skew\_ps, 328  
 vtss\_phy\_sd6g\_ob\_lev\_rd  
   vtss\_phy\_api.h, 945  
 vtss\_phy\_sd6g\_ob\_lev\_wr  
   vtss\_phy\_api.h, 946  
 vtss\_phy\_sd6g\_ob\_post\_rd  
   vtss\_phy\_api.h, 944  
 vtss\_phy\_sd6g\_ob\_post\_wr  
   vtss\_phy\_api.h, 945  
 vtss\_phy\_serdes1g\_rcpll\_status\_get  
   vtss\_phy\_api.h, 943  
 vtss\_phy\_serdes6g\_rcpll\_status\_get  
   vtss\_phy\_api.h, 942  
 vtss\_phy\_serdes\_fmedia\_loopback  
   vtss\_phy\_api.h, 939  
 vtss\_phy\_serdes\_sgmii\_loopback  
   vtss\_phy\_api.h, 939  
 vtss\_phy\_sigdet\_polarity\_t  
   vtss\_phy\_api.h, 901  
 vtss\_phy\_statistic\_get  
   vtss\_phy\_api.h, 931  
 vtss\_phy\_statistic\_t, 328  
   cu\_bad, 329  
   cu\_good, 329  
   media\_mac\_serdes\_crc, 330  
   media\_mac\_serdes\_good, 330  
   rx\_err\_cnt\_base\_tx, 329  
   serdes\_tx\_bad, 329  
   serdes\_tx\_good, 329  
 vtss\_phy\_status\_1g\_get  
   vtss\_phy\_api.h, 913  
 vtss\_phy\_status\_1g\_t, 330  
   master, 331  
   master\_cfg\_fault, 331  
 vtss\_phy\_status\_get  
   vtss\_phy\_api.h, 911  
 vtss\_phy\_status\_inst\_poll  
   vtss\_phy\_api.h, 950  
 vtss\_phy\_tbi\_conf\_t, 331  
   aneg\_enable, 332  
 vtss\_phy\_timestamp\_t, 332  
   high, 332  
   low, 332  
   nanoseconds, 333  
   seconds, 333  
 vtss\_phy\_ts\_10g\_extended\_fifo\_sync  
   vtss\_phy\_ts\_api.h, 1018  
 vtss\_phy\_ts\_10g\_fifo\_sync  
   vtss\_phy\_ts\_api.h, 1018  
 vtss\_phy\_ts\_8487\_xaui\_sel\_t  
   vtss\_phy\_ts\_api.h, 974  
 vtss\_phy\_ts\_ach\_conf\_t, 333  
   channel\_type, 335  
   comm\_opt, 335  
   mask, 334  
     proto\_id, 335  
     value, 334  
     version, 334  
 vtss\_phy\_ts\_action\_format  
   vtss\_phy\_ts\_api.h, 979  
 vtss\_phy\_ts\_alt\_clock\_mode\_get  
   vtss\_phy\_ts\_api.h, 983  
 vtss\_phy\_ts\_alt\_clock\_mode\_s, 335  
   ls\_lpbk, 336  
   ls\_pps\_lpbk, 336  
   pps\_ls\_lpbk, 336  
 vtss\_phy\_ts\_alt\_clock\_mode\_set  
   vtss\_phy\_ts\_api.h, 984  
 vtss\_phy\_ts\_alt\_clock\_mode\_t  
   vtss\_phy\_ts\_api.h, 973  
 vtss\_phy\_ts\_alt\_clock\_saved\_get  
   vtss\_phy\_ts\_api.h, 983  
 vtss\_phy\_ts\_api.h  
   VTSS\_PHY\_TS\_8487\_XAUI\_SEL\_0, 972  
   VTSS\_PHY\_TS\_8487\_XAUI\_SEL\_1, 972  
   VTSS\_PHY\_TS\_DATA\_IN\_RSRVD\_FIELD, 971  
   VTSS\_PHY\_TS\_EGR\_DATAPATH\_RESET, 972  
   VTSS\_PHY\_TS\_EGR\_ENGINE\_ERR, 970  
   VTSS\_PHY\_TS\_EGR\_FIFO\_OVERFLOW, 971  
   VTSS\_PHY\_TS\_EGR\_FIFO\_RESET, 973  
   VTSS\_PHY\_TS\_EGR\_LTC2\_RESET, 973  
   VTSS\_PHY\_TS\_EGR\_RW\_FCS\_ERR, 970  
   VTSS\_PHY\_TS\_EGR\_TIMESTAMP\_CAPTURED, 971  
   VTSS\_PHY\_TS\_ENG\_FLOW\_VALID\_FOR\_CH0, 969  
   VTSS\_PHY\_TS\_ENG\_FLOW\_VALID\_FOR\_CH1, 969  
   VTSS\_PHY\_TS\_ETH\_ADDR\_MATCH\_48BIT, 964  
   VTSS\_PHY\_TS\_ETH\_ADDR\_MATCH\_ANY\_MULTICAST, 965  
   VTSS\_PHY\_TS\_ETH\_ADDR\_MATCH\_ANY\_UNICAST, 965  
   VTSS\_PHY\_TS\_ETH\_MATCH\_DEST\_ADDR, 965  
   VTSS\_PHY\_TS\_ETH\_MATCH\_SRC\_ADDR, 965  
   VTSS\_PHY\_TS\_ETH\_MATCH\_SRC\_OR\_DEST, 965  
   VTSS\_PHY\_TS\_FIFO\_SIG\_DEST\_IP, 961  
   VTSS\_PHY\_TS\_FIFO\_SIG\_DEST\_MAC, 962  
   VTSS\_PHY\_TS\_FIFO\_SIG\_DOMAIN\_NUM, 961  
   VTSS\_PHY\_TS\_FIFO\_SIG\_MSG\_TYPE, 961  
   VTSS\_PHY\_TS\_FIFO\_SIG\_SEQ\_ID, 962  
   VTSS\_PHY\_TS\_FIFO\_SIG\_SOURCE\_PORT\_ID, 962  
   VTSS\_PHY\_TS\_FIFO\_SIG\_SRC\_IP, 961  
   VTSS\_PHY\_TS\_INGR\_DATAPATH\_RESET, 972  
   VTSS\_PHY\_TS\_INGR\_ENGINE\_ERR, 970  
   VTSS\_PHY\_TS\_INGR\_LTC1\_RESET, 972  
   VTSS\_PHY\_TS\_INGR\_RW\_FCS\_ERR, 970  
   VTSS\_PHY\_TS\_INGR\_RW\_PREAM\_ERR, 970  
   VTSS\_PHY\_TS\_IP\_MATCH\_DEST, 968  
   VTSS\_PHY\_TS\_IP\_MATCH\_SRC\_OR\_DEST, 968

VTSS\_PHY\_TS\_IP\_MATCH\_SRC, 967  
VTSS\_PHY\_TS\_IP\_VER\_4, 967  
VTSS\_PHY\_TS\_IP\_VER\_6, 967  
VTSS\_PHY\_TS\_LTC\_LOAD\_SAVE\_NEW\_TOD,  
    971  
VTSS\_PHY\_TS\_LTC\_NEW\_PPS\_INTRPT, 971  
VTSS\_PHY\_TS\_MPLS\_STACK\_DEPTH\_1, 968  
VTSS\_PHY\_TS\_MPLS\_STACK\_DEPTH\_2, 968  
VTSS\_PHY\_TS\_MPLS\_STACK\_DEPTH\_3, 968  
VTSS\_PHY\_TS\_MPLS\_STACK\_DEPTH\_4, 969  
VTSS\_PHY\_TS\_MPLS\_STACK\_REF\_POINT\_←  
    END, 969  
VTSS\_PHY\_TS\_MPLS\_STACK\_REF\_POINT\_←  
    TOP, 969  
VTSS\_PHY\_TS\_SIG\_DEST\_IP\_LEN, 964  
VTSS\_PHY\_TS\_SIG\_DEST\_MAC\_LEN, 964  
VTSS\_PHY\_TS\_SIG\_DOMAIN\_NUM\_LEN, 963  
VTSS\_PHY\_TS\_SIG\_LEN, 962  
VTSS\_PHY\_TS\_SIG\_MSG\_TYPE\_LEN, 963  
VTSS\_PHY\_TS\_SIG\_SEQ\_ID\_LEN, 963  
VTSS\_PHY\_TS\_SIG\_SEQUENCE\_ID\_LEN, 963  
VTSS\_PHY\_TS\_SIG\_SOURCE\_PORT\_ID\_LEN,  
    963  
VTSS\_PHY\_TS\_SIG\_SRC\_IP\_LEN, 964  
VTSS\_PHY\_TS\_SIG\_TIME\_STAMP\_LEN, 962  
VTSS\_PHY\_TS\_TAG\_RANGE\_INNER, 967  
VTSS\_PHY\_TS\_TAG\_RANGE\_NONE, 966  
VTSS\_PHY\_TS\_TAG\_RANGE\_OUTER, 967  
VTSS\_PHY\_TS\_TAG\_TYPE\_B, 966  
VTSS\_PHY\_TS\_TAG\_TYPE\_C, 966  
VTSS\_PHY\_TS\_TAG\_TYPE\_I, 966  
VTSS\_PHY\_TS\_TAG\_TYPE\_S, 966  
VTSS\_PTP\_IP\_1588\_VERSION\_2\_1, 964  
vtss\_phy\_1588\_csr\_reg\_read, 1017  
vtss\_phy\_1588\_csr\_reg\_write, 1016  
vtss\_phy\_1588\_debug\_reg\_read, 1020  
vtss\_phy\_1g\_ts\_fifo\_sync, 1020  
vtss\_phy\_daisy\_conf\_get, 997  
vtss\_phy\_daisy\_conf\_set, 997  
vtss\_phy\_ts\_10g\_extended\_fifo\_sync, 1018  
vtss\_phy\_ts\_10g\_fifo\_sync, 1018  
vtss\_phy\_ts\_8487\_xaui\_sel\_t, 974  
vtss\_phy\_ts\_action\_format, 979  
vtss\_phy\_ts\_alt\_clock\_mode\_get, 983  
vtss\_phy\_ts\_alt\_clock\_mode\_set, 984  
vtss\_phy\_ts\_alt\_clock\_mode\_t, 973  
vtss\_phy\_ts\_alt\_clock\_saved\_get, 983  
vtss\_phy\_ts\_block\_soft\_reset, 1015  
vtss\_phy\_ts\_bypass\_clear, 1018  
vtss\_phy\_ts\_clock\_rateadj\_get, 993  
vtss\_phy\_ts\_clock\_rateadj\_ppm\_get, 994  
vtss\_phy\_ts\_clock\_rateadj\_ppm\_set, 994  
vtss\_phy\_ts\_clock\_rateadj\_set, 993  
vtss\_phy\_ts\_clock\_src\_t, 980  
vtss\_phy\_ts\_clockfreq\_t, 979  
vtss\_phy\_ts\_correction\_overflow\_get, 1012  
vtss\_phy\_ts\_delay\_asymmetry\_get, 989  
vtss\_phy\_ts\_delay\_asymmetry\_set, 988  
vtss\_phy\_ts\_egress\_engine\_action\_get, 1008  
vtss\_phy\_ts\_egress\_engine\_action\_set, 1008  
vtss\_phy\_ts\_egress\_engine\_clear, 1003  
vtss\_phy\_ts\_egress\_engine\_conf\_get, 1005  
vtss\_phy\_ts\_egress\_engine\_conf\_set, 1005  
vtss\_phy\_ts\_egress\_engine\_init, 1002  
vtss\_phy\_ts\_egress\_engine\_init\_conf\_get, 1003  
vtss\_phy\_ts\_egress\_latency\_get, 987  
vtss\_phy\_ts\_egress\_latency\_set, 986  
vtss\_phy\_ts\_encap\_t, 975  
vtss\_phy\_ts\_engine\_channel\_map\_t, 974  
vtss\_phy\_ts\_engine\_flow\_match\_t, 976  
vtss\_phy\_ts\_engine\_t, 976  
vtss\_phy\_ts\_event\_enable\_get, 1010  
vtss\_phy\_ts\_event\_enable\_set, 1010  
vtss\_phy\_ts\_event\_poll, 1011  
vtss\_phy\_ts\_event\_t, 974  
vtss\_phy\_ts\_fifo\_empty, 999  
vtss\_phy\_ts\_fifo\_mode\_t, 981  
vtss\_phy\_ts\_fifo\_read, 974  
vtss\_phy\_ts\_fifo\_read\_cb\_get, 1000  
vtss\_phy\_ts\_fifo\_read\_install, 999  
vtss\_phy\_ts\_fifo\_sig\_get, 998  
vtss\_phy\_ts\_fifo\_sig\_mask\_t, 973  
vtss\_phy\_ts\_fifo\_sig\_set, 998  
vtss\_phy\_ts\_fifo\_status\_t, 975  
vtss\_phy\_ts\_fifo\_timestamp\_len\_t, 981  
vtss\_phy\_ts\_flow\_clear\_cf\_set, 1021  
vtss\_phy\_ts\_hiacc\_get, 1015  
vtss\_phy\_ts\_hiacc\_set, 1014  
vtss\_phy\_ts\_ietf\_mpls\_ach\_oam\_delaym\_type\_t,  
    978  
vtss\_phy\_ts\_ietf\_mpls\_ach\_oam\_ts\_format\_t, 979  
vtss\_phy\_ts\_ingress\_engine\_action\_get, 1006  
vtss\_phy\_ts\_ingress\_engine\_action\_set, 1006  
vtss\_phy\_ts\_ingress\_engine\_clear, 1001  
vtss\_phy\_ts\_ingress\_engine\_conf\_get, 1004  
vtss\_phy\_ts\_ingress\_engine\_conf\_set, 1003  
vtss\_phy\_ts\_ingress\_engine\_init, 1000  
vtss\_phy\_ts\_ingress\_engine\_init\_conf\_get, 1001  
vtss\_phy\_ts\_ingress\_latency\_get, 986  
vtss\_phy\_ts\_ingress\_latency\_set, 985  
vtss\_phy\_ts\_init, 1013  
vtss\_phy\_ts\_init\_conf\_get, 1013  
vtss\_phy\_ts\_load\_ptptime\_get, 991  
vtss\_phy\_ts\_loadpulse\_delay\_get, 993  
vtss\_phy\_ts\_loadpulse\_delay\_set, 992  
vtss\_phy\_ts\_ltc\_freq\_synth\_pulse\_get, 996  
vtss\_phy\_ts\_ltc\_freq\_synth\_pulse\_set, 996  
vtss\_phy\_ts\_mode\_get, 1012  
vtss\_phy\_ts\_mode\_set, 1012  
vtss\_phy\_ts\_nphase\_sampler\_t, 982  
vtss\_phy\_ts\_nphase\_status\_get, 1014  
vtss\_phy\_ts\_ongoing\_adjustment, 996  
vtss\_phy\_ts\_path\_delay\_get, 988  
vtss\_phy\_ts\_path\_delay\_set, 987  
vtss\_phy\_ts\_phy\_oper\_mode\_change, 1016  
vtss\_phy\_ts\_pps\_conf\_get, 985

vtss\_phy\_ts\_pps\_conf\_set, 984  
 vtss\_phy\_ts\_ptp\_clock\_mode\_t, 977  
 vtss\_phy\_ts\_ptp\_delaym\_type\_t, 977  
 vtss\_phy\_ts\_ptp\_message\_type\_t, 982  
 vtss\_phy\_ts\_ptptime\_adj1ns, 995  
 vtss\_phy\_ts\_ptptime\_arm, 990  
 vtss\_phy\_ts\_ptptime\_get, 990  
 vtss\_phy\_ts\_ptptime\_set, 989  
 vtss\_phy\_ts\_ptptime\_set\_done, 990  
 vtss\_phy\_ts\_rxtimestamp\_len\_t, 980  
 vtss\_phy\_ts\_rxtimestamp\_pos\_t, 980  
 vtss\_phy\_ts\_scaled\_ppb\_t, 973  
 vtss\_phy\_ts\_sertod\_get, 992  
 vtss\_phy\_ts\_sertod\_set, 991  
 vtss\_phy\_ts\_soft\_reset\_t, 974  
 vtss\_phy\_ts\_stats\_get, 1011  
 vtss\_phy\_ts\_status\_check, 1017  
 vtss\_phy\_ts\_tc\_op\_mode\_t, 982  
 vtss\_phy\_ts\_tesla\_tsp\_fifo\_sync, 1019  
 vtss\_phy\_ts\_timeofday\_offset\_set, 995  
 vtss\_phy\_ts\_todadj\_status\_t, 975  
 vtss\_phy\_ts\_viper\_fifo\_reset, 1019  
 vtss\_phy\_ts\_y1731\_oam\_delaym\_type\_t, 978  
 vtss\_phy\_ts\_block\_soft\_reset  
     vtss\_phy\_ts\_api.h, 1015  
 vtss\_phy\_ts\_bypass\_clear  
     vtss\_phy\_ts\_api.h, 1018  
 vtss\_phy\_ts\_clock\_rateadj\_get  
     vtss\_phy\_ts\_api.h, 993  
 vtss\_phy\_ts\_clock\_rateadj\_ppm\_get  
     vtss\_phy\_ts\_api.h, 994  
 vtss\_phy\_ts\_clock\_rateadj\_ppm\_set  
     vtss\_phy\_ts\_api.h, 994  
 vtss\_phy\_ts\_clock\_rateadj\_set  
     vtss\_phy\_ts\_api.h, 993  
 vtss\_phy\_ts\_clock\_src\_t  
     vtss\_phy\_ts\_api.h, 980  
 vtss\_phy\_ts\_clockfreq\_t  
     vtss\_phy\_ts\_api.h, 979  
 vtss\_phy\_ts\_correction\_overflow\_get  
     vtss\_phy\_ts\_api.h, 1012  
 vtss\_phy\_ts\_delay\_asymmetry\_get  
     vtss\_phy\_ts\_api.h, 989  
 vtss\_phy\_ts\_delay\_asymmetry\_set  
     vtss\_phy\_ts\_api.h, 988  
 vtss\_phy\_ts\_egress\_engine\_action\_get  
     vtss\_phy\_ts\_api.h, 1008  
 vtss\_phy\_ts\_egress\_engine\_action\_set  
     vtss\_phy\_ts\_api.h, 1008  
 vtss\_phy\_ts\_egress\_engine\_clear  
     vtss\_phy\_ts\_api.h, 1003  
 vtss\_phy\_ts\_egress\_engine\_conf\_get  
     vtss\_phy\_ts\_api.h, 1005  
 vtss\_phy\_ts\_egress\_engine\_conf\_set  
     vtss\_phy\_ts\_api.h, 1005  
 vtss\_phy\_ts\_egress\_engine\_init  
     vtss\_phy\_ts\_api.h, 1002  
 vtss\_phy\_ts\_egress\_engine\_init\_conf\_get  
     vtss\_phy\_ts\_api.h, 1003  
     vtss\_phy\_ts\_egress\_latency\_get  
         vtss\_phy\_ts\_api.h, 987  
     vtss\_phy\_ts\_egress\_latency\_set  
         vtss\_phy\_ts\_api.h, 986  
 vtss\_phy\_ts\_encap\_t  
     vtss\_phy\_ts\_api.h, 975  
 vtss\_phy\_ts\_eng\_init\_conf\_t, 336  
     encap\_type, 337  
     eng\_used, 337  
     flow\_end\_index, 338  
     flow\_match\_mode, 337  
     flow\_st\_index, 337  
 vtss\_phy\_ts\_engine\_action\_t, 338  
     action, 339  
     action\_gen, 339  
     action\_ptp, 339  
     gen\_conf, 339  
     oam\_conf, 339  
     ptp\_conf, 339  
 vtss\_phy\_ts\_engine\_channel\_map\_t  
     vtss\_phy\_ts\_api.h, 974  
 vtss\_phy\_ts\_engine\_flow\_conf\_t, 340  
     channel\_map, 340  
     eng\_mode, 340  
     flow\_conf, 341  
     gen, 341  
     oam, 341  
     ptp, 341  
 vtss\_phy\_ts\_engine\_flow\_match\_t  
     vtss\_phy\_ts\_api.h, 976  
 vtss\_phy\_ts\_engine\_t  
     vtss\_phy\_ts\_api.h, 976  
 vtss\_phy\_ts\_eth\_conf\_t, 342  
     addr\_match\_mode, 344  
     addr\_match\_select, 344  
     comm\_opt, 343  
     etype, 343  
     flow\_en, 343  
     flow\_opt, 347  
     i\_tag, 347  
     inner\_tag, 347  
     inner\_tag\_type, 345  
     lower, 345  
     mac\_addr, 344  
     mask, 346, 347  
     num\_tag, 344  
     outer\_tag, 346  
     outer\_tag\_type, 345  
     pbb\_en, 343  
     range, 346  
     tag\_range\_mode, 345  
     tpid, 343  
     upper, 345  
     val, 346, 347  
     value, 346, 347  
     vlan\_check, 344  
 vtss\_phy\_ts\_event\_enable\_get

vtss\_phy\_ts\_api.h, 1010  
vtss\_phy\_ts\_event\_enable\_set  
    vtss\_phy\_ts\_api.h, 1010  
vtss\_phy\_ts\_event\_poll  
    vtss\_phy\_ts\_api.h, 1011  
vtss\_phy\_ts\_event\_t  
    vtss\_phy\_ts\_api.h, 974  
vtss\_phy\_ts\_fifo\_conf\_t, 348  
    detect\_only, 348  
    eng\_minE, 349  
    eng\_recov, 348  
    skip\_rev\_check, 349  
vtss\_phy\_ts\_fifo\_empty  
    vtss\_phy\_ts\_api.h, 999  
vtss\_phy\_ts\_fifo\_mode\_t  
    vtss\_phy\_ts\_api.h, 981  
vtss\_phy\_ts\_fifo\_read  
    vtss\_phy\_ts\_api.h, 974  
vtss\_phy\_ts\_fifo\_read\_cb\_get  
    vtss\_phy\_ts\_api.h, 1000  
vtss\_phy\_ts\_fifo\_read\_install  
    vtss\_phy\_ts\_api.h, 999  
vtss\_phy\_ts\_fifo\_sig\_get  
    vtss\_phy\_ts\_api.h, 998  
vtss\_phy\_ts\_fifo\_sig\_mask\_t  
    vtss\_phy\_ts\_api.h, 973  
vtss\_phy\_ts\_fifo\_sig\_set  
    vtss\_phy\_ts\_api.h, 998  
vtss\_phy\_ts\_fifo\_sig\_t, 349  
    dest\_ip, 351  
    dest\_mac, 351  
    domain\_num, 350  
    msg\_type, 350  
    sequence\_id, 350  
    sig\_mask, 350  
    src\_ip, 351  
    src\_port\_identity, 350  
vtss\_phy\_ts\_fifo\_status\_t  
    vtss\_phy\_ts\_api.h, 975  
vtss\_phy\_ts\_fifo\_timestamp\_len\_t  
    vtss\_phy\_ts\_api.h, 981  
vtss\_phy\_ts\_flow\_clear\_cf\_set  
    vtss\_phy\_ts\_api.h, 1021  
vtss\_phy\_ts\_gen\_conf\_t, 351  
    comm\_opt, 352  
    data, 353  
    flow\_en, 352  
    flow\_offset, 352  
    flow\_opt, 353  
    mask, 353  
    next\_prot\_offset, 352  
vtss\_phy\_ts\_generic\_action\_t, 353  
    channel\_map, 354  
    data, 354  
    enable, 354  
    flow\_id, 354  
    mask, 355  
    ts\_offset, 355  
        ts\_type, 355  
vtss\_phy\_ts\_generic\_flow\_conf\_t, 355  
    eth1\_opt, 356  
    gen\_opt, 356  
vtss\_phy\_ts\_hiacc\_get  
    vtss\_phy\_ts\_api.h, 1015  
vtss\_phy\_ts\_hiacc\_set  
    vtss\_phy\_ts\_api.h, 1014  
vtss\_phy\_ts\_ietf\_mpls\_ach\_oam\_conf\_t, 356  
    delaym\_type, 357  
    ds, 357  
    ts\_format, 357  
vtss\_phy\_ts\_ietf\_mpls\_ach\_oam\_delaym\_type\_t  
    vtss\_phy\_ts\_api.h, 978  
vtss\_phy\_ts\_ietf\_mpls\_ach\_oam\_ts\_format\_t  
    vtss\_phy\_ts\_api.h, 979  
vtss\_phy\_ts\_ingress\_engine\_action\_get  
    vtss\_phy\_ts\_api.h, 1006  
vtss\_phy\_ts\_ingress\_engine\_action\_set  
    vtss\_phy\_ts\_api.h, 1006  
vtss\_phy\_ts\_ingress\_engine\_clear  
    vtss\_phy\_ts\_api.h, 1001  
vtss\_phy\_ts\_ingress\_engine\_conf\_get  
    vtss\_phy\_ts\_api.h, 1004  
vtss\_phy\_ts\_ingress\_engine\_conf\_set  
    vtss\_phy\_ts\_api.h, 1003  
vtss\_phy\_ts\_ingress\_engine\_init  
    vtss\_phy\_ts\_api.h, 1000  
vtss\_phy\_ts\_ingress\_engine\_init\_conf\_get  
    vtss\_phy\_ts\_api.h, 1001  
vtss\_phy\_ts\_ingress\_latency\_get  
    vtss\_phy\_ts\_api.h, 986  
vtss\_phy\_ts\_ingress\_latency\_set  
    vtss\_phy\_ts\_api.h, 985  
vtss\_phy\_ts\_init  
    vtss\_phy\_ts\_api.h, 1013  
vtss\_phy\_ts\_init\_conf\_get  
    vtss\_phy\_ts\_api.h, 1013  
vtss\_phy\_ts\_init\_conf\_t, 357  
    auto\_clear\_ls, 360  
    chk\_ing\_modified, 361  
    clk\_freq, 358  
    clk\_src, 358  
    macsec\_ena, 360  
    one\_step\_txfifo, 361  
    rx\_ts\_len, 359  
    rx\_ts\_pos, 358  
    tc\_op\_mode, 360  
    tx\_fifo\_hi\_clk\_cycs, 359  
    tx\_fifo\_lo\_clk\_cycs, 360  
    tx\_fifo\_mode, 359  
    tx\_fifo\_spi\_conf, 359  
    tx\_ts\_len, 359  
    xauisel\_8487, 360  
vtss\_phy\_ts\_ip\_conf\_t, 361  
    addr, 364  
    comm\_opt, 363  
    dport\_mask, 363

dport\_val, 363  
 flow\_en, 363  
 flow\_opt, 365  
 ip\_addr, 365  
 ip\_mode, 362  
 ipv4, 364  
 ipv6, 364  
 mask, 364  
 match\_mode, 364  
 sport\_mask, 363  
 sport\_val, 362  
 vtss\_phy\_ts\_load\_ptptime\_get  
     vtss\_phy\_ts\_api.h, 991  
 vtss\_phy\_ts\_loadpulse\_delay\_get  
     vtss\_phy\_ts\_api.h, 993  
 vtss\_phy\_ts\_loadpulse\_delay\_set  
     vtss\_phy\_ts\_api.h, 992  
 vtss\_phy\_ts\_ltc\_freq\_synth\_pulse\_get  
     vtss\_phy\_ts\_api.h, 996  
 vtss\_phy\_ts\_ltc\_freq\_synth\_pulse\_set  
     vtss\_phy\_ts\_api.h, 996  
 vtss\_phy\_ts\_mode\_get  
     vtss\_phy\_ts\_api.h, 1012  
 vtss\_phy\_ts\_mode\_set  
     vtss\_phy\_ts\_api.h, 1012  
 vtss\_phy\_ts\_mpls\_conf\_t, 365  
     bottom\_up, 368  
     comm\_opt, 366  
     cw\_en, 366  
     end, 368  
     flow\_en, 366  
     flow\_opt, 369  
     frst\_lvl\_after\_top, 367  
     frst\_lvl\_before\_end, 368  
     snd\_lvl\_after\_top, 367  
     snd\_lvl\_before\_end, 368  
     stack\_depth, 366  
     stack\_level, 368  
     stack\_ref\_point, 366  
     thrd\_lvl\_after\_top, 367  
     thrd\_lvl\_before\_end, 368  
     top, 367  
     top\_down, 367  
 vtss\_phy\_ts\_mpls\_lvl\_rng\_t, 369  
     lower, 369  
     upper, 369  
 vtss\_phy\_ts\_nphase\_sampler\_t  
     vtss\_phy\_ts\_api.h, 982  
 vtss\_phy\_ts\_nphase\_status\_get  
     vtss\_phy\_ts\_api.h, 1014  
 vtss\_phy\_ts\_nphase\_status\_t, 370  
     CALIB\_DONE, 371  
     CALIB\_ERR, 370  
     enable, 370  
 vtss\_phy\_ts\_oam\_engine\_action\_t, 371  
     channel\_map, 372  
     enable, 372  
     ietf\_oam\_conf, 372  
         oam\_conf, 373  
         version, 372  
         y1731\_en, 372  
         y1731\_oam\_conf, 372  
 vtss\_phy\_ts\_oam\_engine\_flow\_conf\_t, 373  
     ach\_opt, 374  
     eth1\_opt, 373  
     eth2\_opt, 374  
     mpls\_opt, 374  
 vtss\_phy\_ts\_ongoing\_adjustment  
     vtss\_phy\_ts\_api.h, 996  
 vtss\_phy\_ts\_path\_delay\_get  
     vtss\_phy\_ts\_api.h, 988  
 vtss\_phy\_ts\_path\_delay\_set  
     vtss\_phy\_ts\_api.h, 987  
 vtss\_phy\_ts\_phy\_oper\_mode\_change  
     vtss\_phy\_ts\_api.h, 1016  
 vtss\_phy\_ts\_pps\_conf\_get  
     vtss\_phy\_ts\_api.h, 985  
 vtss\_phy\_ts\_pps\_conf\_set  
     vtss\_phy\_ts\_api.h, 984  
 vtss\_phy\_ts\_pps\_config\_s, 374  
     pps\_offset, 375  
     pps\_output\_enable, 375  
     pps\_width\_adj, 375  
 vtss\_phy\_ts\_ptp\_clock\_mode\_t  
     vtss\_phy\_ts\_api.h, 977  
 vtss\_phy\_ts\_ptp\_conf\_t, 376  
     domain, 377  
     lower, 377  
     mask, 376  
     range, 377  
     range\_en, 376  
     upper, 377  
     val, 376  
     value, 377  
 vtss\_phy\_ts\_ptp\_delaym\_type\_t  
     vtss\_phy\_ts\_api.h, 977  
 vtss\_phy\_ts\_ptp\_engine\_action\_t, 378  
     cf\_update, 379  
     channel\_map, 378  
     clk\_mode, 379  
     delaym\_type, 379  
     enable, 378  
     ptp\_conf, 378  
 vtss\_phy\_ts\_ptp\_engine\_flow\_conf\_t, 379  
     ach\_opt, 381  
     eth1\_opt, 380  
     eth2\_opt, 380  
     ip1\_opt, 380  
     ip2\_opt, 380  
     mpls\_opt, 381  
 vtss\_phy\_ts\_ptp\_message\_type\_t  
     vtss\_phy\_ts\_api.h, 982  
 vtss\_phy\_ts\_ptptime\_adj1ns  
     vtss\_phy\_ts\_api.h, 995  
 vtss\_phy\_ts\_ptptime\_arm  
     vtss\_phy\_ts\_api.h, 990

vtss\_phy\_ts\_ptptime\_get  
    vtss\_phy\_ts\_api.h, 990  
vtss\_phy\_ts\_ptptime\_set  
    vtss\_phy\_ts\_api.h, 989  
vtss\_phy\_ts\_ptptime\_set\_done  
    vtss\_phy\_ts\_api.h, 990  
vtss\_phy\_ts\_rxtimestamp\_len\_t  
    vtss\_phy\_ts\_api.h, 980  
vtss\_phy\_ts\_rxtimestamp\_pos\_t  
    vtss\_phy\_ts\_api.h, 980  
vtss\_phy\_ts\_scaled\_ppb\_t  
    vtss\_phy\_ts\_api.h, 973  
vtss\_phy\_ts\_sertod\_conf\_t, 381  
    ip\_enable, 382  
    ls\_inv, 382  
    op\_enable, 382  
vtss\_phy\_ts\_sertod\_get  
    vtss\_phy\_ts\_api.h, 992  
vtss\_phy\_ts\_sertod\_set  
    vtss\_phy\_ts\_api.h, 991  
vtss\_phy\_ts\_soft\_reset\_t  
    vtss\_phy\_ts\_api.h, 974  
vtss\_phy\_ts\_stats\_get  
    vtss\_phy\_ts\_api.h, 1011  
vtss\_phy\_ts\_stats\_t, 382  
    egr\_fcs\_err, 383  
    egr\_frm\_mod\_cnt, 384  
    egr\_pream\_shrink\_err, 383  
    ingr\_fcs\_err, 383  
    ingr\_frm\_mod\_cnt, 384  
    ingr\_pream\_shrink\_err, 383  
    ts\_fifo\_drop\_cnt, 384  
    ts\_fifo\_tx\_cnt, 384  
vtss\_phy\_ts\_status\_check  
    vtss\_phy\_ts\_api.h, 1017  
vtss\_phy\_ts\_tc\_op\_mode\_t  
    vtss\_phy\_ts\_api.h, 982  
vtss\_phy\_ts\_tesla\_tsp\_fifo\_sync  
    vtss\_phy\_ts\_api.h, 1019  
vtss\_phy\_ts\_timeofday\_offset\_set  
    vtss\_phy\_ts\_api.h, 995  
vtss\_phy\_ts\_todadj\_status\_t  
    vtss\_phy\_ts\_api.h, 975  
vtss\_phy\_ts\_viper\_fifo\_reset  
    vtss\_phy\_ts\_api.h, 1019  
vtss\_phy\_ts\_y1731\_oam\_conf\_t, 385  
    delaym\_type, 385  
    lower, 386  
    mask, 386  
    meg\_level, 386  
    range, 386  
    range\_en, 385  
    upper, 386  
    val, 385  
    value, 386  
vtss\_phy\_ts\_y1731\_oam\_delaym\_type\_t  
    vtss\_phy\_ts\_api.h, 978  
vtss\_phy\_type\_t, 387  
base\_port\_no, 388  
channel\_id, 388  
part\_number, 387  
phy\_api\_base\_no, 388  
port\_cnt, 388  
revision, 387  
vtss\_phy\_unidirectional\_t  
    vtss\_phy\_api.h, 902  
vtss\_phy\_veriphy\_get  
    vtss\_phy\_api.h, 922  
vtss\_phy\_veriphy\_result\_t, 389  
    length, 389  
    link, 389  
    status, 389  
vtss\_phy\_veriphy\_start  
    vtss\_phy\_api.h, 922  
vtss\_phy\_veriphy\_status\_t  
    phy.h, 555  
vtss\_phy\_warm\_start\_10g\_failed\_get  
    vtss\_phy\_10g\_api.h, 874  
vtss\_phy\_warm\_start\_failed\_get  
    vtss\_phy\_api.h, 937  
vtss\_phy\_wol\_conf\_get  
    vtss\_phy\_api.h, 941  
vtss\_phy\_wol\_conf\_set  
    vtss\_phy\_api.h, 941  
vtss\_phy\_wol\_conf\_t, 390  
    magic\_pkt\_cnt, 391  
    secure\_on\_enable, 390  
    wol\_mac, 390  
    wol\_pass, 391  
    wol\_passwd\_len, 391  
vtss\_phy\_wol\_enable  
    vtss\_phy\_api.h, 940  
vtss\_phy\_write  
    vtss\_phy\_api.h, 919  
vtss\_phy\_write\_masked  
    vtss\_phy\_api.h, 919  
vtss\_phy\_write\_masked\_page  
    vtss\_phy\_api.h, 920  
vtss\_pi\_conf\_t, 391  
    cs\_wait\_ns, 392  
vtss\_policer\_ext\_t, 392  
    cpu\_queue, 395  
    dp\_bypass\_level, 393  
    flow\_control, 395  
    frame\_rate, 393  
    known\_broadcast, 393  
    known\_multicast, 393  
    known\_unicast, 393  
    learning, 394  
    limit\_cpu\_traffic, 395  
    limit\_noncpu\_traffic, 395  
    to\_cpu, 394  
    unknown\_broadcast, 394  
    unknown\_multicast, 394  
    unknown\_unicast, 394  
vtss\_policer\_t, 396

level, 396  
 rate, 396  
**vtss\_policer\_type\_t**  
 types.h, 602  
**vtss\_poll\_1sec**  
 vtss\_misc\_api.h, 721  
**vtss\_port\_api.h**  
 CHIP\_PORT\_UNUSED, 1024  
 VTSS\_FRAME\_GAP\_DEFAULT, 1024  
 VTSS\_MAX\_FRAME\_LENGTH\_MAX, 1025  
 VTSS\_MAX\_FRAME\_LENGTH\_STANDARD,  
     1025  
 vtss\_internal\_bw\_t, 1026  
 vtss\_miim\_controller\_t, 1025  
 vtss\_miim\_read, 1034  
 vtss\_miim\_write, 1034  
 vtss\_mmd\_read, 1037  
 vtss\_mmd\_write, 1038  
 vtss\_port\_basic\_counters\_get, 1032  
 vtss\_port\_clause\_37\_control\_get, 1028  
 vtss\_port\_clause\_37\_control\_set, 1029  
 vtss\_port\_clause\_37\_remote\_fault\_t, 1026  
 vtss\_port\_conf\_get, 1030  
 vtss\_port\_conf\_set, 1029  
 vtss\_port\_counters\_clear, 1031  
 vtss\_port\_counters\_get, 1031  
 vtss\_port\_counters\_update, 1030  
 vtss\_port\_forward\_state\_get, 1032  
 vtss\_port\_forward\_state\_set, 1033  
 vtss\_port\_forward\_t, 1027  
 vtss\_port\_ifh\_conf\_get, 1033  
 vtss\_port\_ifh\_conf\_set, 1033  
 vtss\_port\_loop\_t, 1027  
 vtss\_port\_map\_get, 1028  
 vtss\_port\_map\_set, 1028  
 vtss\_port\_max\_tags\_t, 1026  
 vtss\_port\_mmd\_masked\_write, 1036  
 vtss\_port\_mmd\_read, 1035  
 vtss\_port\_mmd\_read\_inc, 1035  
 vtss\_port\_mmd\_write, 1036  
 vtss\_port\_status\_get, 1030  
**vtss\_port\_basic\_counters\_get**  
 vtss\_port\_api.h, 1032  
**vtss\_port\_bridge\_counters\_t**, 396  
 dot1dTpPortInDiscards, 397  
**vtss\_port\_clause\_37\_adv\_t**, 397  
 acknowledge, 398  
 asymmetric\_pause, 398  
 fdx, 398  
 hdx, 398  
 next\_page, 399  
 remote\_fault, 398  
 symmetric\_pause, 398  
**vtss\_port\_clause\_37\_control\_get**  
 vtss\_port\_api.h, 1028  
**vtss\_port\_clause\_37\_control\_set**  
 vtss\_port\_api.h, 1029  
**vtss\_port\_clause\_37\_control\_t**, 399  
 advertisement, 400  
 enable, 399  
**vtss\_port\_clause\_37\_remote\_fault\_t**  
 vtss\_port\_api.h, 1026  
**vtss\_port\_conf\_get**  
 vtss\_port\_api.h, 1030  
**vtss\_port\_conf\_set**  
 vtss\_port\_api.h, 1029  
**vtss\_port\_conf\_t**, 400  
 exc\_col\_cont, 403  
 fdx, 402  
 flow\_control, 402  
 frame\_gaps, 401  
 frame\_length\_chk, 403  
 if\_type, 401  
 loop, 404  
 max\_frame\_length, 402  
 max\_tags, 403  
 power\_down, 402  
 sd\_active\_high, 401  
 sd\_enable, 401  
 sd\_internal, 401  
 serdes, 404  
 speed, 402  
 xau\_rx\_lane\_flip, 403  
 xau\_tx\_lane\_flip, 403  
**vtss\_port\_counters\_clear**  
 vtss\_port\_api.h, 1031  
**vtss\_port\_counters\_get**  
 vtss\_port\_api.h, 1031  
**vtss\_port\_counters\_t**, 404  
 bridge, 405  
 ethernet\_like, 405  
 if\_group, 405  
 prop, 405  
 rmon, 405  
**vtss\_port\_counters\_update**  
 vtss\_port\_api.h, 1030  
**vtss\_port\_ethernet\_like\_counters\_t**, 406  
 dot3ControllnUnknownOpcodes, 407  
 dot3InPauseFrames, 407  
 dot3OutPauseFrames, 409  
 dot3StatsAlignmentErrors, 406  
 dot3StatsCarrierSenseErrors, 409  
 dot3StatsDeferredTransmissions, 408  
 dot3StatsExcessiveCollisions, 408  
 dot3StatsFCSErrors, 407  
 dot3StatsFrameTooLongs, 407  
 dot3StatsLateCollisions, 408  
 dot3StatsMultipleCollisionFrames, 408  
 dot3StatsSingleCollisionFrames, 408  
 dot3StatsSymbolErrors, 407  
**vtss\_port\_flow\_control\_conf\_t**, 409  
 generate, 410  
 obey, 410  
 pfc, 410  
 smac, 410  
**vtss\_port\_forward\_state\_get**

vtss\_port\_api.h, 1032  
vtss\_port\_forward\_state\_set  
    vtss\_port\_api.h, 1033  
vtss\_port\_forward\_t  
    vtss\_port\_api.h, 1027  
vtss\_port\_frame\_gaps\_t, 411  
    fdx\_gap, 411  
    hdx\_gap\_1, 411  
    hdx\_gap\_2, 411  
vtss\_port\_if\_group\_counters\_t, 412  
    ifInBroadcastPkts, 413  
    ifInDiscards, 413  
    ifInErrors, 414  
    ifInMulticastPkts, 413  
    ifInNUcastPkts, 413  
    ifInOctets, 412  
    ifInUcastPkts, 413  
    ifOutBroadcastPkts, 414  
    ifOutDiscards, 415  
    ifOutErrors, 415  
    ifOutMulticastPkts, 414  
    ifOutNUcastPkts, 415  
    ifOutOctets, 414  
    ifOutUcastPkts, 414  
vtss\_port\_ifh\_conf\_get  
    vtss\_port\_api.h, 1033  
vtss\_port\_ifh\_conf\_set  
    vtss\_port\_api.h, 1033  
vtss\_port\_ifh\_t, 415  
    ena\_ifh\_header, 416  
    ena\_inj\_header, 416  
    ena\_xtr\_header, 416  
vtss\_port\_interface\_t  
    types.h, 600  
vtss\_port\_loop\_t  
    vtss\_port\_api.h, 1027  
vtss\_port\_map\_get  
    vtss\_port\_api.h, 1028  
vtss\_port\_map\_set  
    vtss\_port\_api.h, 1028  
vtss\_port\_map\_t, 417  
    chip\_no, 417  
    chip\_port, 417  
    max\_bw, 417  
    miim\_addr, 418  
    miim\_chip\_no, 418  
    miim\_controller, 418  
vtss\_port\_max\_tags\_t  
    vtss\_port\_api.h, 1026  
vtss\_port\_mmd\_masked\_write  
    vtss\_port\_api.h, 1036  
vtss\_port\_mmd\_read  
    vtss\_port\_api.h, 1035  
vtss\_port\_mmd\_read\_inc  
    vtss\_port\_api.h, 1035  
vtss\_port\_mmd\_write  
    vtss\_port\_api.h, 1036  
vtss\_port\_mux\_mode\_t  
    vtss\_init\_api.h, 632  
vtss\_port\_proprietary\_counters\_t, 418  
    rx\_prio, 419  
    tx\_prio, 419  
vtss\_port\_rmon\_counters\_t, 419  
    rx\_etherStatsBroadcastPkts, 421  
    rx\_etherStatsCRCAlignErrors, 421  
    rx\_etherStatsDropEvents, 420  
    rx\_etherStatsFragments, 422  
    rx\_etherStatsJabbers, 422  
    rx\_etherStatsMulticastPkts, 421  
    rx\_etherStatsOctets, 420  
    rx\_etherStatsOversizePkts, 422  
    rx\_etherStatsPkts, 421  
    rx\_etherStatsPkts1024to1518Octets, 423  
    rx\_etherStatsPkts128to255Octets, 423  
    rx\_etherStatsPkts1519toMaxOctets, 423  
    rx\_etherStatsPkts256to511Octets, 423  
    rx\_etherStatsPkts512to1023Octets, 423  
    rx\_etherStatsPkts64Octets, 422  
    rx\_etherStatsPkts65to127Octets, 422  
    rx\_etherStatsUndersizePkts, 421  
    tx\_etherStatsBroadcastPkts, 424  
    tx\_etherStatsCollisions, 425  
    tx\_etherStatsDropEvents, 424  
    tx\_etherStatsMulticastPkts, 424  
    tx\_etherStatsOctets, 424  
    tx\_etherStatsPkts, 424  
    tx\_etherStatsPkts1024to1518Octets, 426  
    tx\_etherStatsPkts128to255Octets, 425  
    tx\_etherStatsPkts1519toMaxOctets, 426  
    tx\_etherStatsPkts256to511Octets, 425  
    tx\_etherStatsPkts512to1023Octets, 426  
    tx\_etherStatsPkts64Octets, 425  
    tx\_etherStatsPkts65to127Octets, 425  
vtss\_port\_serdes\_conf\_t, 426  
    sfp\_dac, 427  
vtss\_port\_sgmii\_aneg\_t, 427  
    aneg\_complete, 429  
    fdx, 428  
    hdx, 428  
    link, 428  
    speed\_100M, 428  
    speed\_10M, 428  
    speed\_1G, 428  
vtss\_port\_speed\_t  
    port.h, 568  
vtss\_port\_state\_get  
    vtss\_l2\_api.h, 665  
vtss\_port\_state\_set  
    vtss\_l2\_api.h, 666  
vtss\_port\_status\_get  
    vtss\_port\_api.h, 1030  
vtss\_port\_status\_t, 429  
    aneg, 431  
    aneg\_complete, 430  
    copper, 431  
    fdx, 430

fiber, 431  
 link, 430  
 link\_down, 430  
 mdi\_cross, 431  
 remote\_fault, 430  
 speed, 430  
 unidirectional\_ability, 431  
 vtss\_ptp\_event\_enable  
     vtss\_misc\_api.h, 722  
 vtss\_ptp\_event\_poll  
     vtss\_misc\_api.h, 721  
 vtss\_pvlan\_port\_members\_get  
     vtss\_l2\_api.h, 680  
 vtss\_pvlan\_port\_members\_set  
     vtss\_l2\_api.h, 680  
 vtss\_qce\_action\_t, 432  
     dei, 434  
     dp, 433  
     dp\_enable, 433  
     dscp, 433  
     dscp\_enable, 433  
     pcp, 434  
     pcp\_dei\_enable, 433  
     policy\_no, 434  
     policy\_no\_enable, 434  
     prio, 432  
     prio\_enable, 432  
 vtss\_qce\_add  
     vtss\_qos\_api.h, 1046  
 vtss\_qce\_del  
     vtss\_qos\_api.h, 1047  
 vtss\_qce\_frame\_etype\_t, 435  
     data, 435  
     etype, 435  
 vtss\_qce\_frame\_ipv4\_t, 435  
     dip, 437  
     dport, 437  
     dscp, 436  
     fragment, 436  
     proto, 436  
     sip, 436  
     sport, 437  
 vtss\_qce\_frame\_ipv6\_t, 437  
     dip, 438  
     dport, 439  
     dscp, 438  
     proto, 438  
     sip, 438  
     sport, 439  
 vtss\_qce\_frame\_llc\_t, 439  
     data, 440  
 vtss\_qce\_frame\_snap\_t, 440  
     data, 440  
 vtss\_qce\_init  
     vtss\_qos\_api.h, 1046  
 vtss\_qce\_key\_t, 441  
     etype, 442  
     frame, 443  
     inner\_tag, 442  
     ipv4, 443  
     ipv6, 443  
     llc, 442  
     mac, 441  
     port\_list, 441  
     snap, 442  
     tag, 441  
     type, 442  
 vtss\_qce\_mac\_t, 443  
     dmac, 444  
     dmac\_bc, 444  
     dmac\_mc, 444  
     smac, 444  
 vtss\_qce\_t, 445  
     action, 446  
     id, 445  
     key, 445  
 vtss\_qce\_tag\_t, 446  
     dei, 447  
     pcp, 447  
     s\_tag, 447  
     tagged, 447  
     vid, 446  
 vtss\_qce\_type\_t  
     vtss\_qos\_api.h, 1044  
 vtss\_qos\_api.h  
     VTSS\_PORT\_POLICER\_CPU\_QUEUES, 1041  
     VTSS\_PORT\_POLICERS, 1041  
     VTSS\_QCE\_ID\_LAST, 1042  
     VTSS\_QCL\_ARRAY\_SIZE, 1042  
     VTSS\_QCL\_ID\_END, 1042  
     VTSS\_QCL\_ID\_START, 1041  
     VTSS\_QCL\_IDS, 1041  
     VTSS\_WRED\_DPL\_CNT, 1040  
     VTSS\_WRED\_GROUP\_CNT, 1041  
     vtss\_dscp\_emode\_t, 1043  
     vtss\_dscp\_mode\_t, 1043  
     vtss\_qce\_add, 1046  
     vtss\_qce\_del, 1047  
     vtss\_qce\_init, 1046  
     vtss\_qce\_type\_t, 1044  
     vtss\_qos\_conf\_get, 1044  
     vtss\_qos\_conf\_set, 1045  
     vtss\_qos\_port\_conf\_get, 1045  
     vtss\_qos\_port\_conf\_set, 1045  
     vtss\_red\_v3\_t, 1042  
     vtss\_tag\_remark\_mode\_t, 1043  
     vtss\_wred\_v2\_max\_t, 1042  
 vtss\_qos\_conf\_get  
     vtss\_qos\_api.h, 1044  
 vtss\_qos\_conf\_set  
     vtss\_qos\_api.h, 1045  
 vtss\_qos\_conf\_t, 448  
     dscp\_dp\_level\_map, 449  
     dscp\_qos\_class\_map, 449  
     dscp\_qos\_map, 449  
     dscp\_qos\_map\_dp1, 449

dscp\_qos\_map\_dp2, 449  
dscp\_qos\_map\_dp3, 450  
dscp\_remap, 450  
dscp\_remark, 450  
dscp\_translate\_map, 450  
dscp\_trust, 448  
policer\_bc, 452  
policer\_bc\_frame\_rate, 452  
policer\_bc\_mode, 452  
policer\_mc, 451  
policer\_mc\_frame\_rate, 451  
policer\_mc\_mode, 451  
policer\_uc, 450  
policer\_uc\_frame\_rate, 451  
policer\_uc\_mode, 451  
prios, 448  
red\_v3, 452  
vtss\_qos\_port\_conf\_get  
    vtss\_qos\_api.h, 1045  
vtss\_qos\_port\_conf\_set  
    vtss\_qos\_api.h, 1045  
vtss\_qos\_port\_conf\_t, 453  
    default\_dei, 455  
    default\_dpl, 455  
    default\_prio, 454  
    dp\_level\_map, 455  
    dscp\_class\_enable, 456  
    dscp\_emode, 456  
    dscp\_mode, 456  
    dscp\_translate, 456  
    dwrr\_cnt, 458  
    dwrr\_enable, 457  
    policer\_ext\_port, 453  
    policer\_port, 453  
    policer\_queue, 454  
    qos\_class\_map, 455  
    queue\_pct, 458  
    shaper\_port, 454  
    shaper\_queue, 454  
    tag\_class\_enable, 455  
    tag\_default\_dei, 457  
    tag\_default\_pcp, 457  
    tag\_dei\_map, 457  
    tag\_pcp\_map, 457  
    tag\_remark\_mode, 456  
    usr\_prio, 454  
    wred\_group, 458  
vtss\_rcpll\_status\_t, 458  
    cal\_error, 459  
    cal\_not\_done, 459  
    out\_of\_range, 459  
vtss\_recvrd\_clkout\_t  
    vtss\_phy\_10g\_api.h, 818  
vtss\_recvrd\_t  
    vtss\_phy\_10g\_api.h, 814  
vtss\_recvrdclk\_cdr\_div\_t  
    vtss\_phy\_10g\_api.h, 814  
vtss\_red\_v2\_t, 460  
enable, 460  
max, 460  
max\_unit, 461  
min\_fl, 460  
vtss\_red\_v3\_t  
    vtss\_qos\_api.h, 1042  
vtss\_reg\_read  
    vtss\_misc\_api.h, 719  
vtss\_reg\_read\_t  
    vtss\_init\_api.h, 626  
vtss\_reg\_write  
    vtss\_misc\_api.h, 719  
vtss\_reg\_write\_masked  
    vtss\_misc\_api.h, 719  
vtss\_reg\_write\_t  
    vtss\_init\_api.h, 626  
vtss\_restart\_conf\_end  
    vtss\_init\_api.h, 635  
vtss\_restart\_conf\_get  
    vtss\_init\_api.h, 636  
vtss\_restart\_conf\_set  
    vtss\_init\_api.h, 636  
vtss\_restart\_status\_get  
    vtss\_init\_api.h, 636  
vtss\_restart\_status\_t, 461  
    cur\_version, 462  
    prev\_version, 462  
    restart, 461  
vtss\_restart\_t  
    vtss\_init\_api.h, 633  
vtss\_routing\_entry\_t, 462  
    ipv4\_uc, 463  
    ipv6\_uc, 463  
    route, 463  
    type, 463  
    vlan, 463  
vtss\_rptr\_rate\_t  
    vtss\_phy\_10g\_api.h, 816  
vtss\_rx\_master\_timestamp\_get  
    vtss\_ts\_api.h, 1080  
vtss\_rx\_timestamp\_get  
    vtss\_ts\_api.h, 1079  
vtss\_rx\_timestamp\_id\_release  
    vtss\_ts\_api.h, 1080  
vtss\_secure\_on\_passwd\_t, 464  
    passwd, 464  
vtss\_security\_api.h  
    VTSS\_ACE\_ID\_LAST, 1050  
    vtss\_ace\_add, 1057  
    vtss\_ace\_bit\_t, 1053  
    vtss\_ace\_counter\_clear, 1058  
    vtss\_ace\_counter\_get, 1058  
    vtss\_ace\_del, 1057  
    vtss\_ace\_init, 1056  
    vtss\_ace\_type\_t, 1052  
    vtss\_acl\_policer\_conf\_get, 1054  
    vtss\_acl\_policer\_conf\_set, 1054  
    vtss\_acl\_port\_action\_t, 1051

vtss\_acl\_port\_conf\_get, 1055  
 vtss\_acl\_port\_conf\_set, 1055  
 vtss\_acl\_port\_counter\_clear, 1056  
 vtss\_acl\_port\_counter\_get, 1056  
 vtss\_acl\_ptp\_action\_t, 1051  
 vtss\_acl\_ptp\_rsp\_t, 1052  
 vtss\_auth\_port\_state\_get, 1053  
 vtss\_auth\_port\_state\_set, 1053  
 vtss\_auth\_state\_t, 1051  
 vtss\_serdes\_macro\_conf\_t, 465  
     ib\_cterm\_ena, 465  
     qsgmii, 466  
     serdes1g\_vdd, 465  
     serdes6g\_vdd, 465  
 vtss\_serdes\_mode\_t  
     types.h, 601  
 vtss\_sflow\_port\_conf\_get  
     vtss\_l2\_api.h, 682  
 vtss\_sflow\_port\_conf\_set  
     vtss\_l2\_api.h, 683  
 vtss\_sflow\_port\_conf\_t, 466  
     sampling\_rate, 467  
     type, 466  
 vtss\_sflow\_sampling\_rate\_convert  
     vtss\_l2\_api.h, 683  
 vtss\_sflow\_type\_t  
     l2\_types.h, 531  
 vtss\_sgpiobmode\_t  
     vtss\_misc\_api.h, 713  
 vtss\_sgpiocfg\_get  
     vtss\_misc\_api.h, 726  
 vtss\_sgpiocfg\_set  
     vtss\_misc\_api.h, 726  
 vtss\_sgpiocfg\_t, 467  
     bit\_count, 468  
     bmode, 467  
     port\_conf, 468  
 vtss\_sgpioevent\_enable  
     vtss\_misc\_api.h, 728  
 vtss\_sgpioevent\_poll  
     vtss\_misc\_api.h, 727  
 vtss\_sgpiemode\_t  
     vtss\_misc\_api.h, 713  
 vtss\_sgpioportconf\_t, 468  
     enabled, 469  
     int\_pol\_high, 469  
     mode, 469  
 vtss\_sgpioportdata\_t, 469  
     value, 470  
 vtss\_sgpioread  
     vtss\_misc\_api.h, 727  
 vtss\_shaper\_t, 470  
     ebs, 471  
     eir, 471  
     level, 470  
     rate, 471  
 vtss\_spidev32bitreadwrite\_t  
     vtss\_init\_api.h, 628  
     vtss\_spi\_64bit\_read\_write\_t  
         vtss\_init\_api.h, 629  
 vtss\_spireadwrite\_t  
     vtss\_init\_api.h, 628  
 vtss\_squelch\_workaround  
     vtss\_phy\_api.h, 930  
 vtss\_srefclkdiv\_t  
     vtss\_phy\_10g\_api.h, 814  
 vtss\_stormpolicer\_mode\_t  
     types.h, 602  
 vtss\_stpportstate\_get  
     vtss\_l2\_api.h, 666  
 vtss\_stpportstate\_set  
     vtss\_l2\_api.h, 666  
 vtss\_stpstate\_t  
     vtss\_l2\_api.h, 653  
 vtss\_sublayerstatus\_t, 471  
     link\_down, 472  
     rx\_fault, 472  
     rx\_link, 472  
     tx\_fault, 472  
 vtss\_syncapi.h  
     VTSS\_SYNCE\_CLK\_MAX, 1060  
     VTSS\_SYNCE\_CLK\_A, 1060  
     VTSS\_SYNCE\_CLK\_B, 1060  
     vtss\_sync\_clock\_in\_get, 1062  
     vtss\_sync\_clock\_in\_set, 1062  
     vtss\_sync\_clock\_out\_get, 1061  
     vtss\_sync\_clock\_out\_set, 1061  
     vtss\_sync\_divider\_t, 1060  
 vtss\_sync\_clock\_in\_get  
     vtss\_syncapi.h, 1062  
 vtss\_sync\_clock\_in\_set  
     vtss\_syncapi.h, 1062  
 vtss\_sync\_clockint\_t, 473  
     enable, 474  
     port\_no, 473  
     squelsh, 473  
 vtss\_sync\_clockout\_get  
     vtss\_syncapi.h, 1061  
 vtss\_sync\_clockout\_set  
     vtss\_syncapi.h, 1061  
 vtss\_sync\_clockout\_t, 474  
     divider, 474  
     enable, 475  
 vtss\_syncdivider\_t  
     vtss\_syncapi.h, 1060  
 vtss\_tagremarkmode\_t  
     vtss\_qosapi.h, 1043  
 vtss\_tagtype\_t  
     vtss\_packetapi.h, 767  
 vtss\_targettype\_t  
     vtss\_initapi.h, 632  
 vtss\_tci\_t, 475  
     cfi, 476  
     tagprio, 476  
     vid, 475  
 vtss\_tempsensor\_get

vtss\_misc\_api.h, 732  
vtss\_temp\_sensor\_init  
    vtss\_misc\_api.h, 732  
vtss\_timeofday\_t, 476  
    sec, 477  
vtss\_timestamp\_age  
    vtss\_ts\_api.h, 1081  
vtss\_timestamp\_t, 477  
    nanoseconds, 478  
    sec\_msb, 478  
    seconds, 478  
vtss\_tod\_get\_ns\_cnt  
    vtss\_misc\_api.h, 731  
vtss\_tod\_set\_ns\_cnt\_cb  
    vtss\_misc\_api.h, 732  
vtss\_trace\_conf\_get  
    vtss\_misc\_api.h, 714  
vtss\_trace\_conf\_set  
    vtss\_misc\_api.h, 715  
vtss\_trace\_conf\_t, 478  
    level, 479  
vtss\_trace\_group\_t  
    vtss\_misc\_api.h, 710  
vtss\_trace\_layer\_t  
    vtss\_misc\_api.h, 710  
vtss\_trace\_level\_t  
    vtss\_misc\_api.h, 711  
vtss\_ts\_adjtimer\_get  
    vtss\_ts\_api.h, 1070  
vtss\_ts\_adjtimer\_one\_sec  
    vtss\_ts\_api.h, 1068  
vtss\_ts\_adjtimer\_set  
    vtss\_ts\_api.h, 1069  
vtss\_ts\_alt\_clock\_mode\_get  
    vtss\_ts\_api.h, 1072  
vtss\_ts\_alt\_clock\_mode\_set  
    vtss\_ts\_api.h, 1072  
vtss\_ts\_alt\_clock\_mode\_t, 479  
    load, 480  
    one\_pps\_in, 480  
    one\_pps\_out, 479  
    save, 480  
    vtss\_ts\_api.h, 1066  
vtss\_ts\_alt\_clock\_saved\_get  
    vtss\_ts\_api.h, 1070  
vtss\_ts\_api.h  
    vtss\_rx\_master\_timestamp\_get, 1080  
    vtss\_rx\_timestamp\_get, 1079  
    vtss\_rx\_timestamp\_id\_release, 1080  
    vtss\_timestamp\_age, 1081  
    vtss\_ts\_adjtimer\_get, 1070  
    vtss\_ts\_adjtimer\_one\_sec, 1068  
    vtss\_ts\_adjtimer\_set, 1069  
    vtss\_ts\_alt\_clock\_mode\_get, 1072  
    vtss\_ts\_alt\_clock\_mode\_set, 1072  
    vtss\_ts\_alt\_clock\_mode\_t, 1066  
    vtss\_ts\_alt\_clock\_saved\_get, 1070  
    vtss\_ts\_delay\_asymmetry\_get, 1077  
vtss\_ts\_delay\_asymmetry\_set, 1077  
vtss\_ts\_egress\_latency\_get, 1076  
vtss\_ts\_egress\_latency\_set, 1076  
vtss\_ts\_external\_clock\_mode\_get, 1073  
vtss\_ts\_external\_clock\_mode\_set, 1073  
vtss\_ts\_external\_clock\_saved\_get, 1074  
vtss\_ts\_freq\_offset\_get, 1070  
vtss\_ts\_ingress\_latency\_get, 1075  
vtss\_ts\_ingress\_latency\_set, 1074  
vtss\_ts\_internal\_mode\_get, 1079  
vtss\_ts\_internal\_mode\_set, 1078  
vtss\_ts\_ongoing\_adjustment, 1068  
vtss\_ts\_operation\_mode\_get, 1078  
vtss\_ts\_operation\_mode\_set, 1077  
vtss\_ts\_p2p\_delay\_get, 1075  
vtss\_ts\_p2p\_delay\_set, 1075  
vtss\_ts\_status\_change, 1081  
vtss\_ts\_timeofday\_get, 1068  
vtss\_ts\_timeofday\_next\_pps\_get, 1069  
vtss\_ts\_timeofday\_next\_pps\_set, 1072  
vtss\_ts\_timeofday\_offset\_set, 1067  
vtss\_ts\_timeofday\_set, 1066  
vtss\_ts\_timeofday\_set\_delta, 1067  
vtss\_tx\_timestamp\_idx\_alloc, 1081  
vtss\_tx\_timestamp\_update, 1079  
vtss\_ts\_delay\_asymmetry\_get  
    vtss\_ts\_api.h, 1077  
vtss\_ts\_delay\_asymmetry\_set  
    vtss\_ts\_api.h, 1077  
vtss\_ts\_egress\_latency\_get  
    vtss\_ts\_api.h, 1076  
vtss\_ts\_egress\_latency\_set  
    vtss\_ts\_api.h, 1076  
vtss\_ts\_ext\_clock\_mode\_t, 480  
    enable, 481  
    freq, 481  
    one\_pps\_mode, 481  
vtss\_ts\_external\_clock\_mode\_get  
    vtss\_ts\_api.h, 1073  
vtss\_ts\_external\_clock\_mode\_set  
    vtss\_ts\_api.h, 1073  
vtss\_ts\_external\_clock\_saved\_get  
    vtss\_ts\_api.h, 1074  
vtss\_ts\_freq\_offset\_get  
    vtss\_ts\_api.h, 1070  
vtss\_ts\_id\_t, 482  
    ts\_id, 482  
vtss\_ts\_ingress\_latency\_get  
    vtss\_ts\_api.h, 1075  
vtss\_ts\_ingress\_latency\_set  
    vtss\_ts\_api.h, 1074  
vtss\_ts\_internal\_mode\_get  
    vtss\_ts\_api.h, 1079  
vtss\_ts\_internal\_mode\_set  
    vtss\_ts\_api.h, 1078  
vtss\_ts\_internal\_mode\_t, 482  
    int\_fmt, 483  
vtss\_ts\_ongoing\_adjustment

vtss\_ts\_api.h, 1068  
 vtss\_ts\_operation\_mode\_get  
     vtss\_ts\_api.h, 1078  
 vtss\_ts\_operation\_mode\_set  
     vtss\_ts\_api.h, 1077  
 vtss\_ts\_operation\_mode\_t, 483  
     mode, 483  
 vtss\_ts\_p2p\_delay\_get  
     vtss\_ts\_api.h, 1075  
 vtss\_ts\_p2p\_delay\_set  
     vtss\_ts\_api.h, 1075  
 vtss\_ts\_status\_change  
     vtss\_ts\_api.h, 1081  
 vtss\_ts\_timeofday\_get  
     vtss\_ts\_api.h, 1068  
 vtss\_ts\_timeofday\_next\_pps\_get  
     vtss\_ts\_api.h, 1069  
 vtss\_ts\_timeofday\_next\_pps\_set  
     vtss\_ts\_api.h, 1072  
 vtss\_ts\_timeofday\_offset\_set  
     vtss\_ts\_api.h, 1067  
 vtss\_ts\_timeofday\_set  
     vtss\_ts\_api.h, 1066  
 vtss\_ts\_timeofday\_set\_delta  
     vtss\_ts\_api.h, 1067  
 vtss\_ts\_timestamp\_alloc\_t, 484  
     cb, 485  
     context, 484  
     port\_mask, 484  
 vtss\_ts\_timestamp\_t, 485  
     context, 486  
     id, 486  
     ts, 485  
     ts\_valid, 486  
 vtss\_tx\_timestamp\_idx\_alloc  
     vtss\_ts\_api.h, 1081  
 vtss\_tx\_timestamp\_update  
     vtss\_ts\_api.h, 1079  
 vtss\_uc\_flood\_members\_get  
     vtss\_l2\_api.h, 692  
 vtss\_uc\_flood\_members\_set  
     vtss\_l2\_api.h, 693  
 vtss\_vcap\_bit\_t  
     types.h, 603  
 vtss\_vcap\_ip\_t, 486  
     mask, 487  
     value, 487  
 vtss\_vcap\_key\_type\_t  
     types.h, 604  
 vtss\_vcap\_u128\_t, 487  
     mask, 488  
     value, 488  
 vtss\_vcap\_u16\_t, 488  
     mask, 489  
     value, 489  
 vtss\_vcap\_u24\_t, 489  
     mask, 490  
     value, 490  
 vtss\_vcap\_u32\_t, 490  
     mask, 491  
     value, 491  
 vtss\_vcap\_u40\_t, 491  
     mask, 492  
     value, 492  
 vtss\_vcap\_u48\_t, 492  
     mask, 493  
     value, 493  
 vtss\_vcap\_u8\_t, 493  
     mask, 494  
     value, 494  
 vtss\_vcap\_udp\_tcp\_t, 494  
     high, 495  
     in\_range, 495  
     low, 495  
 vtss\_vcap\_vid\_t, 495  
     mask, 496  
     value, 496  
 vtss\_vcap\_vr\_t, 496  
     high, 498  
     low, 498  
     mask, 497  
     r, 498  
     type, 497  
     v, 498  
     value, 497  
     vr, 498  
 vtss\_vcap\_vr\_type\_t  
     types.h, 604  
 vtss\_vce\_action\_t, 499  
     policy\_no, 499  
     vid, 499  
 vtss\_vce\_add  
     vtss\_l2\_api.h, 675  
 vtss\_vce\_del  
     vtss\_l2\_api.h, 675  
 vtss\_vce\_frame\_etype\_t, 499  
     data, 500  
     etype, 500  
 vtss\_vce\_frame\_ipv4\_t, 500  
     dport, 502  
     dscp, 501  
     fragment, 501  
     options, 501  
     proto, 501  
     sip, 502  
 vtss\_vce\_frame\_ip6\_t, 502  
     dport, 503  
     dscp, 503  
     proto, 503  
     sip, 503  
 vtss\_vce\_frame\_llc\_t, 504  
     data, 504  
 vtss\_vce\_frame\_snap\_t, 504  
     data, 505  
 vtss\_vce\_init  
     vtss\_l2\_api.h, 675

vtss\_vce\_key\_t, 505  
etype, 506  
frame, 507  
ipv4, 507  
ipv6, 507  
llc, 506  
mac, 506  
port\_list, 506  
snap, 507  
tag, 506  
type, 506  
vtss\_vce\_mac\_t, 508  
dmac\_bc, 508  
dmac\_mc, 508  
smac, 508  
vtss\_vce\_t, 509  
action, 510  
id, 509  
key, 509  
vtss\_vce\_tag\_t, 510  
dei, 511  
pcp, 511  
s\_tag, 511  
tagged, 511  
vid, 510  
vtss\_vce\_type\_t  
    vtss\_l2\_api.h, 654  
vtss\_vcl\_port\_conf\_get  
    vtss\_l2\_api.h, 674  
vtss\_vcl\_port\_conf\_set  
    vtss\_l2\_api.h, 674  
vtss\_vcl\_port\_conf\_t, 512  
    dmac\_dip, 512  
vtss\_vdd\_t  
    types.h, 603  
vtss\_vid\_mac\_t, 512  
    mac, 513  
    vid, 513  
vtss\_vlan\_conf\_get  
    vtss\_l2\_api.h, 669  
vtss\_vlan\_conf\_set  
    vtss\_l2\_api.h, 669  
vtss\_vlan\_conf\_t, 513  
    s\_etype, 514  
vtss\_vlan\_frame\_t  
    types.h, 602  
vtss\_vlan\_port\_conf\_get  
    vtss\_l2\_api.h, 669  
vtss\_vlan\_port\_conf\_set  
    vtss\_l2\_api.h, 671  
vtss\_vlan\_port\_conf\_t, 514  
    frame\_type, 515  
    ingress\_filter, 515  
    port\_type, 514  
    pvid, 514  
    untagged\_vid, 515  
vtss\_vlan\_port\_members\_get  
    vtss\_l2\_api.h, 671  
                vtss\_vlan\_port\_members\_set  
                vtss\_l2\_api.h, 672  
                vtss\_vlan\_port\_type\_t  
                vtss\_l2\_api.h, 654  
                vtss\_vlan\_tag\_t, 515  
                dei, 516  
                pcp, 516  
                tpid, 516  
                vid, 516  
                vtss\_vlan\_trans\_group\_add  
                vtss\_l2\_api.h, 676  
                vtss\_vlan\_trans\_group\_del  
                vtss\_l2\_api.h, 676  
                vtss\_vlan\_trans\_group\_get  
                vtss\_l2\_api.h, 677  
                vtss\_vlan\_trans\_group\_to\_port\_get  
                vtss\_l2\_api.h, 678  
                vtss\_vlan\_trans\_group\_to\_port\_set  
                vtss\_l2\_api.h, 677  
                vtss\_vlan\_trans\_grp2vlan\_conf\_t, 517  
                group\_id, 517  
                trans\_vid, 518  
                vid, 517  
                vtss\_vlan\_trans\_port2grp\_conf\_t, 518  
                group\_id, 518  
                ports, 519  
                vtss\_vlan\_tx\_tag\_get  
                vtss\_l2\_api.h, 673  
                vtss\_vlan\_tx\_tag\_set  
                vtss\_l2\_api.h, 673  
                vtss\_vlan\_tx\_tag\_t  
                vtss\_l2\_api.h, 654  
                vtss\_vlan\_vid\_conf\_get  
                vtss\_l2\_api.h, 672  
                vtss\_vlan\_vid\_conf\_set  
                vtss\_l2\_api.h, 672  
                vtss\_vlan\_vid\_conf\_t, 519  
                learning, 519  
                mirror, 520  
                vtss\_vstax\_conf\_get  
                vtss\_l2\_api.h, 699  
                vtss\_vstax\_conf\_set  
                vtss\_l2\_api.h, 700  
                vtss\_vstax\_conf\_t, 520  
                cmef\_disable, 521  
                port\_0, 521  
                port\_1, 521  
                upsid\_0, 520  
                upsid\_1, 521  
                vtss\_vstax\_fwd\_mode\_t  
                vtss\_packet\_api.h, 766  
                vtss\_vstax\_glag\_entry\_t, 522  
                upsid, 522  
                upspn, 522  
                vtss\_vstax\_glag\_get  
                vtss\_l2\_api.h, 687  
                vtss\_vstax\_glag\_set  
                vtss\_l2\_api.h, 687

vtss\_vstax\_master\_upsid\_get  
     vtss\_l2\_api.h, 701  
 vtss\_vstax\_master\_upsid\_set  
     vtss\_l2\_api.h, 702  
 vtss\_vstax\_port\_conf\_get  
     vtss\_l2\_api.h, 700  
 vtss\_vstax\_port\_conf\_set  
     vtss\_l2\_api.h, 701  
 vtss\_vstax\_port\_conf\_t, 522  
     mirror, 523  
     ttl, 523  
 vtss\_vstax\_route\_entry\_t, 523  
     stack\_port\_a, 524  
     stack\_port\_b, 524  
 vtss\_vstax\_route\_table\_t, 524  
     table, 525  
     topology\_type, 525  
 vtss\_vstax\_rx\_header\_t, 525  
     glag\_no, 526  
     isdx, 527  
     port\_no, 526  
     sp, 526  
     upsid, 526  
     valid, 526  
 vtss\_vstax\_topology\_set  
     vtss\_l2\_api.h, 702  
 vtss\_vstax\_topology\_type\_t  
     vtss\_l2\_api.h, 656  
 vtss\_vstax\_tx\_header\_t, 527  
     chip\_port, 529  
     dp, 529  
     fwd\_mode, 528  
     glag\_no, 529  
     keep\_ttl, 529  
     port\_no, 528  
     prio, 528  
     queue\_no, 529  
     tci, 528  
     ttl, 528  
     upsid, 528  
 vtss\_vt\_id\_t  
     vtss\_l2\_api.h, 653  
 vtss\_wis\_api.h  
     VTSS\_EWIS\_AISL\_EV, 1089  
     VTSS\_EWIS\_AISP\_EV, 1090  
     VTSS\_EWIS\_B1\_NZ\_EV, 1092  
     VTSS\_EWIS\_B1\_THRESH\_EV, 1093  
     VTSS\_EWIS\_B2\_NZ\_EV, 1092  
     VTSS\_EWIS\_B2\_THRESH\_EV, 1093  
     VTSS\_EWIS\_B3\_NZ\_EV, 1093  
     VTSS\_EWIS\_B3\_THRESH\_EV, 1094  
     VTSS\_EWIS\_FAIS\_EV, 1088  
     VTSS\_EWIS\_FERDIP\_EV, 1091  
     VTSS\_EWIS\_FEUNEQP\_EV, 1091  
     VTSS\_EWIS\_FPLM\_EV, 1088  
     VTSS\_EWIS\_HIGH\_BER\_EV, 1092  
     VTSS\_EWIS\_LCDP\_EV, 1089  
     VTSS\_EWIS\_LOF\_EV, 1088  
     VTSS\_EWIS\_LOPC\_EV, 1091  
     VTSS\_EWIS\_LOPP\_EV, 1090  
     VTSS\_EWIS\_LOS\_EV, 1089  
     VTSS\_EWIS\_MODULE\_EV, 1090  
     VTSS\_EWIS\_PCS\_RECEIVE\_FAULT\_PEND, 1092  
     VTSS\_EWIS\_PLMP\_EV, 1089  
     VTSS\_EWIS\_RDIL\_EV, 1089  
     VTSS\_EWIS\_REIL\_EV, 1091  
     VTSS\_EWIS\_REIL\_NZ\_EV, 1093  
     VTSS\_EWIS\_REIL\_THRESH\_EV, 1094  
     VTSS\_EWIS\_REIP\_EV, 1092  
     VTSS\_EWIS\_REIP\_NZ\_EV, 1093  
     VTSS\_EWIS\_REIP\_THRESH\_EV, 1094  
     VTSS\_EWIS\_RXLOL\_EV, 1090  
     VTSS\_EWIS\_SEF\_EV, 1088  
     VTSS\_EWIS\_TXLOL\_EV, 1090  
     VTSS\_EWIS\_UNEQP\_EV, 1091  
     vtss\_ewis\_cons\_act\_get, 1103  
     vtss\_ewis\_cons\_act\_set, 1103  
     vtss\_ewis\_counter\_get, 1110  
     vtss\_ewis\_counter\_threshold\_get, 1111  
     vtss\_ewis\_counter\_threshold\_set, 1111  
     vtss\_ewis\_defects\_get, 1108  
     vtss\_ewis\_event\_enable, 1096  
     vtss\_ewis\_event\_force, 1098  
     vtss\_ewis\_event\_poll, 1097  
     vtss\_ewis\_event\_poll\_without\_mask, 1098  
     vtss\_ewis\_event\_t, 1094  
     vtss\_ewis\_exp\_sl\_set, 1105  
     vtss\_ewis\_force\_conf\_get, 1099  
     vtss\_ewis\_force\_conf\_set, 1099  
     vtss\_ewis\_mode\_get, 1102  
     vtss\_ewis\_mode\_set, 1102  
     vtss\_ewis\_mode\_t, 1095  
     vtss\_ewis\_path\_acti\_get, 1109  
     vtss\_ewis\_path\_txti\_get, 1106  
     vtss\_ewis\_path\_txti\_set, 1105  
     vtss\_ewis\_perf\_cntr\_mode\_t, 1095  
     vtss\_ewis\_perf\_get, 1110  
     vtss\_ewis\_perf\_mode\_get, 1112  
     vtss\_ewis\_perf\_mode\_set, 1112  
     vtss\_ewis\_prbs31\_err\_inj\_set, 1107  
     vtss\_ewis\_prbs31\_err\_inj\_t, 1096  
     vtss\_ewis\_reset, 1103  
     vtss\_ewis\_section\_acti\_get, 1109  
     vtss\_ewis\_section\_txti\_get, 1104  
     vtss\_ewis\_section\_txti\_set, 1104  
     vtss\_ewis\_static\_conf\_get, 1099  
     vtss\_ewis\_static\_conf\_t, 1094  
     vtss\_ewis\_status\_get, 1109  
     vtss\_ewis\_test\_counter\_get, 1108  
     vtss\_ewis\_test\_mode\_get, 1107  
     vtss\_ewis\_test\_mode\_set, 1106  
     vtss\_ewis\_test\_pattern\_s, 1096  
     vtss\_ewis\_tti\_mode\_t, 1095  
     vtss\_ewis\_tx\_oh\_get, 1100  
     vtss\_ewis\_tx\_oh\_passthru\_get, 1101

vtss\_ewis\_tx\_oh\_passthru\_set, 1101  
vtss\_ewis\_tx\_oh\_set, 1100  
vtss\_wol\_mac\_addr\_t, 530  
    addr, 530  
vtss\_wol\_passwd\_len\_type\_t  
    vtss\_phy\_api.h, 905  
vtss\_wred\_v2\_max\_t  
    vtss\_qos\_api.h, 1042  
vtss\_wref\_clk\_div\_t  
    vtss\_phy\_10g\_api.h, 815  
vtss\_wrefclk\_t  
    vtss\_phy\_10g\_api.h, 813

warm\_start\_enable  
    vtss\_init\_conf\_t, 138

wis  
    vtss\_phy\_10g\_serdess\_status\_t, 290  
    vtss\_phy\_10g\_status\_t, 292

wol\_mac  
    vtss\_phy\_wol\_conf\_t, 390

wol\_pass  
    vtss\_phy\_wol\_conf\_t, 391

wol\_passwd\_len  
    vtss\_phy\_wol\_conf\_t, 391

wred\_group  
    vtss\_qos\_port\_conf\_t, 458

wref\_clk\_div  
    vtss\_phy\_10g\_mode\_t, 255

wrefclk  
    vtss\_phy\_10g\_mode\_t, 253

x\_count  
    vtss\_phy\_10g\_vscope\_scan\_conf\_t, 298

x\_incr  
    vtss\_phy\_10g\_vscope\_scan\_conf\_t, 298

x\_start  
    vtss\_phy\_10g\_vscope\_scan\_conf\_t, 297

xaui\_lane\_flip  
    vtss\_phy\_10g\_mode\_t, 254

xaui\_rx\_lane\_flip  
    vtss\_port\_conf\_t, 403

xaui\_sel\_8487  
    vtss\_phy\_ts\_init\_conf\_t, 360

xaui\_tx\_lane\_flip  
    vtss\_port\_conf\_t, 403

xfi\_pol\_invert  
    vtss\_phy\_10g\_mode\_t, 254

xs  
    vtss\_phy\_10g\_status\_t, 293

xtr\_cb  
    vtss\_fdma\_ch\_cfg\_t, 128

xtr\_grp  
    vtss\_fdma\_ch\_cfg\_t, 127

xtr\_qu  
    vtss\_packet\_rx\_meta\_t, 184

xtr\_qu\_mask  
    vtss\_packet\_rx\_info\_t, 177

y1731\_en