

# Vitesse API

Generated by Doxygen 1.8.14



# Contents

<b>1</b>	<b>Ethernet Virtual Connections</b>	<b>1</b>
1.1	Port Configuration . . . . .	1
1.2	Policer Configuration . . . . .	1
1.3	EVCs . . . . .	2
1.4	ECEs . . . . .	2
1.5	EVC Port Statistics . . . . .	2
<b>2</b>	<b>Layer 2</b>	<b>3</b>
2.1	MAC Address Table . . . . .	3
2.2	Operational State . . . . .	4
2.3	Spanning Tree . . . . .	4
2.4	VLAN . . . . .	4
2.5	VLAN Classification List . . . . .	5
2.6	VLAN Translation . . . . .	5
2.7	Port Isolation . . . . .	6
2.8	Private VLAN . . . . .	6
2.9	Asymmetric Private VLAN . . . . .	6
2.10	Destination Port Groups . . . . .	6
2.11	sFlow . . . . .	7
2.12	Link Aggregation . . . . .	7
2.13	Mirroring . . . . .	7
2.14	Flooding Control . . . . .	8
2.15	IPv4 Multicast . . . . .	8
2.16	IPv6 Multicast . . . . .	8
2.17	Ethernet Protection Switching . . . . .	9
2.18	Ethernet Ring Protection Switching . . . . .	9

<b>3 Security</b>	<b>11</b>
3.1 Port Authentication (802.1X) . . . . .	11
3.2 Access Control List . . . . .	11
3.2.1 Access Control Entry . . . . .	11
3.2.2 Port Configuration . . . . .	12
3.2.3 Policer Configuration . . . . .	12
<b>4 Data Structure Index</b>	<b>13</b>
4.1 Data Structures . . . . .	13
<b>5 File Index</b>	<b>21</b>
5.1 File List . . . . .	21
<b>6 Data Structure Documentation</b>	<b>23</b>
6.1 port_custom_conf_t Struct Reference . . . . .	23
6.1.1 Detailed Description . . . . .	23
6.1.2 Field Documentation . . . . .	23
6.1.2.1 enable . . . . .	24
6.1.2.2 autoneg . . . . .	24
6.1.2.3 fdx . . . . .	24
6.1.2.4 flow_control . . . . .	24
6.1.2.5 pfc . . . . .	24
6.1.2.6 speed . . . . .	25
6.1.2.7 dual_media_fiber_speed . . . . .	25
6.1.2.8 max_length . . . . .	25
6.1.2.9 exc_col_cont . . . . .	25
6.1.2.10 adv_dis . . . . .	25
6.1.2.11 max_tags . . . . .	26
6.1.2.12 oper_up . . . . .	26
6.1.2.13 frame_length_chk . . . . .	26
6.2 serdes_fields_t Struct Reference . . . . .	26
6.2.1 Detailed Description . . . . .	26

6.2.2	Field Documentation	27
6.2.2.1	ob_post0	27
6.2.2.2	ob_sr	27
6.3	vtss_ace_frame_arp_t Struct Reference	27
6.3.1	Detailed Description	28
6.3.2	Field Documentation	28
6.3.2.1	smac	28
6.3.2.2	arp	28
6.3.2.3	req	28
6.3.2.4	unknown	28
6.3.2.5	smac_match	29
6.3.2.6	dmac_match	29
6.3.2.7	length	29
6.3.2.8	ip	29
6.3.2.9	ethernet	29
6.3.2.10	sip	30
6.3.2.11	dip	30
6.4	vtss_ace_frame_etype_t Struct Reference	30
6.4.1	Detailed Description	30
6.4.2	Field Documentation	30
6.4.2.1	dmac	31
6.4.2.2	smac	31
6.4.2.3	etype	31
6.4.2.4	data	31
6.4.2.5	ptp	31
6.5	vtss_ace_frame_ipv4_t Struct Reference	32
6.5.1	Detailed Description	32
6.5.2	Field Documentation	32
6.5.2.1	ttl	32
6.5.2.2	fragment	33

6.5.2.3	options	33
6.5.2.4	ds	33
6.5.2.5	proto	33
6.5.2.6	sip	33
6.5.2.7	dip	34
6.5.2.8	data	34
6.5.2.9	sport	34
6.5.2.10	dport	34
6.5.2.11	tcp_fin	34
6.5.2.12	tcp_syn	35
6.5.2.13	tcp_RST	35
6.5.2.14	tcp_psh	35
6.5.2.15	tcp_ack	35
6.5.2.16	tcp_urg	35
6.5.2.17	sip_eq_dip	36
6.5.2.18	sport_eq_dport	36
6.5.2.19	seq_zero	36
6.5.2.20	ptp	36
6.5.2.21	sip_smac	36
6.6	vtss_ace_frame_ipv6_t Struct Reference	37
6.6.1	Detailed Description	37
6.6.2	Field Documentation	37
6.6.2.1	proto	37
6.6.2.2	sip	38
6.6.2.3	ttl	38
6.6.2.4	ds	38
6.6.2.5	data	38
6.6.2.6	sport	38
6.6.2.7	dport	39
6.6.2.8	tcp_fin	39

6.6.2.9	tcp_syn	39
6.6.2.10	tcp_RST	39
6.6.2.11	tcp_PUSH	39
6.6.2.12	tcp_ACK	40
6.6.2.13	tcp_URG	40
6.6.2.14	sip_EQ_DIP	40
6.6.2.15	sport_EQ_dport	40
6.6.2.16	seq_zero	40
6.6.2.17	ptp	41
6.7	vtss_ace_frame_llc_t Struct Reference	41
6.7.1	Detailed Description	41
6.7.2	Field Documentation	41
6.7.2.1	dmac	41
6.7.2.2	smac	42
6.7.2.3	llc	42
6.8	vtss_ace_frame_snap_t Struct Reference	42
6.8.1	Detailed Description	42
6.8.2	Field Documentation	42
6.8.2.1	dmac	43
6.8.2.2	smac	43
6.8.2.3	snap	43
6.9	vtss_ace_ptp_t Struct Reference	43
6.9.1	Detailed Description	43
6.9.2	Field Documentation	44
6.9.2.1	enable	44
6.9.2.2	header	44
6.10	vtss_ace_sip_smac_t Struct Reference	44
6.10.1	Detailed Description	44
6.10.2	Field Documentation	45
6.10.2.1	enable	45

---

6.10.2.2	sip	45
6.10.2.3	smac	45
6.11	vtss_ace_t Struct Reference	45
6.11.1	Detailed Description	46
6.11.2	Field Documentation	46
6.11.2.1	id	46
6.11.2.2	port_list	46
6.11.2.3	policy	47
6.11.2.4	type	47
6.11.2.5	action	47
6.11.2.6	dmac_mc	47
6.11.2.7	dmac_bc	47
6.11.2.8	vlan	48
6.11.2.9	etype	48
6.11.2.10	llc	48
6.11.2.11	snap	48
6.11.2.12	arp	48
6.11.2.13	ipv4	49
6.11.2.14	ipv6	49
6.11.2.15	frame	49
6.12	vtss_ace_vlan_t Struct Reference	49
6.12.1	Detailed Description	49
6.12.2	Field Documentation	50
6.12.2.1	vid	50
6.12.2.2	usr_prio	50
6.12.2.3	cfi	50
6.12.2.4	tagged	50
6.13	vtss_acl_action_t Struct Reference	51
6.13.1	Detailed Description	51
6.13.2	Field Documentation	51

---

6.13.2.1	cpu	51
6.13.2.2	cpu_once	51
6.13.2.3	cpu_queue	52
6.13.2.4	police	52
6.13.2.5	policer_no	52
6.13.2.6	evc_police	52
6.13.2.7	evc_policer_id	52
6.13.2.8	learn	53
6.13.2.9	port_action	53
6.13.2.10	port_list	53
6.13.2.11	mirror	53
6.13.2.12	ptp_action	53
6.14	vtss_acl_policer_conf_t Struct Reference	54
6.14.1	Detailed Description	54
6.14.2	Field Documentation	54
6.14.2.1	bit_rate_enable	54
6.14.2.2	bit_rate	54
6.14.2.3	rate	55
6.15	vtss_acl_port_conf_t Struct Reference	55
6.15.1	Detailed Description	55
6.15.2	Field Documentation	55
6.15.2.1	policy_no	55
6.15.2.2	action	56
6.16	vtss_aggr_mode_t Struct Reference	56
6.16.1	Detailed Description	56
6.16.2	Field Documentation	56
6.16.2.1	smac_enable	56
6.16.2.2	dmac_enable	57
6.16.2.3	sip_dip_enable	57
6.16.2.4	sport_dport_enable	57

6.17 vtss_aneg_t Struct Reference . . . . .	57
6.17.1 Detailed Description . . . . .	57
6.17.2 Field Documentation . . . . .	58
6.17.2.1 obey_pause . . . . .	58
6.17.2.2 generate_pause . . . . .	58
6.18 vtss_api_lock_t Struct Reference . . . . .	58
6.18.1 Detailed Description . . . . .	58
6.18.2 Field Documentation . . . . .	59
6.18.2.1 inst . . . . .	59
6.18.2.2 function . . . . .	59
6.18.2.3 file . . . . .	59
6.18.2.4 line . . . . .	59
6.19 vtss_basic_counters_t Struct Reference . . . . .	60
6.19.1 Detailed Description . . . . .	60
6.19.2 Field Documentation . . . . .	60
6.19.2.1 rx_frames . . . . .	60
6.19.2.2 tx_frames . . . . .	60
6.20 vtss_chip_id_t Struct Reference . . . . .	60
6.20.1 Detailed Description . . . . .	61
6.20.2 Field Documentation . . . . .	61
6.20.2.1 part_number . . . . .	61
6.20.2.2 revision . . . . .	61
6.21 vtss_counter_pair_t Struct Reference . . . . .	61
6.21.1 Detailed Description . . . . .	62
6.21.2 Field Documentation . . . . .	62
6.21.2.1 frames . . . . .	62
6.21.2.2 bytes . . . . .	62
6.22 vtss_debug_info_t Struct Reference . . . . .	62
6.22.1 Detailed Description . . . . .	63
6.22.2 Field Documentation . . . . .	63

---

6.22.2.1	layer	63
6.22.2.2	group	63
6.22.2.3	chip_no	63
6.22.2.4	port_list	63
6.22.2.5	full	64
6.22.2.6	clear	64
6.22.2.7	vml_format	64
6.23	vtss_debug_lock_t Struct Reference	64
6.23.1	Detailed Description	64
6.23.2	Field Documentation	65
6.23.2.1	chip_no	65
6.24	vtss_dgroup_port_conf_t Struct Reference	65
6.24.1	Detailed Description	65
6.24.2	Field Documentation	65
6.24.2.1	dgroup_no	65
6.25	vtss_dlb_policer_conf_t Struct Reference	66
6.25.1	Detailed Description	66
6.25.2	Field Documentation	66
6.25.2.1	type	66
6.25.2.2	enable	66
6.25.2.3	cf	67
6.25.2.4	line_rate	67
6.25.2.5	cir	67
6.25.2.6	cbs	67
6.25.2.7	eir	67
6.25.2.8	ebs	68
6.26	vtss_ece_action_t Struct Reference	68
6.26.1	Detailed Description	68
6.26.2	Field Documentation	68
6.26.2.1	dir	68

---

6.26.2.2	pop_tag	69
6.26.2.3	outer_tag	69
6.26.2.4	evc_id	69
6.26.2.5	policy_no	69
6.26.2.6	prio_enable	69
6.26.2.7	prio	70
6.27	vtss_ece_frame_ipv4_t Struct Reference	70
6.27.1	Detailed Description	70
6.27.2	Field Documentation	70
6.27.2.1	dscp	70
6.27.2.2	fragment	71
6.27.2.3	proto	71
6.27.2.4	sip	71
6.27.2.5	sport	71
6.27.2.6	dport	71
6.28	vtss_ece_frame_ipv6_t Struct Reference	72
6.28.1	Detailed Description	72
6.28.2	Field Documentation	72
6.28.2.1	dscp	72
6.28.2.2	proto	72
6.28.2.3	sip	73
6.28.2.4	sport	73
6.28.2.5	dport	73
6.29	vtss_ece_key_t Struct Reference	73
6.29.1	Detailed Description	74
6.29.2	Field Documentation	74
6.29.2.1	port_list	74
6.29.2.2	mac	74
6.29.2.3	tag	74
6.29.2.4	type	74

6.29.2.5  ipv4 . . . . .	75
6.29.2.6  ipv6 . . . . .	75
6.29.2.7  frame . . . . .	75
6.30  vtss_ece_mac_t Struct Reference . . . . .	75
6.30.1  Detailed Description . . . . .	75
6.30.2  Field Documentation . . . . .	76
6.30.2.1  dmac_mc . . . . .	76
6.30.2.2  dmac_bc . . . . .	76
6.30.2.3  smac . . . . .	76
6.31  vtss_ece_outer_tag_t Struct Reference . . . . .	76
6.31.1  Detailed Description . . . . .	77
6.31.2  Field Documentation . . . . .	77
6.31.2.1  enable . . . . .	77
6.31.2.2  pcp_dei_preserve . . . . .	77
6.31.2.3  pcp . . . . .	77
6.31.2.4  dei . . . . .	77
6.32  vtss_ece_t Struct Reference . . . . .	78
6.32.1  Detailed Description . . . . .	78
6.32.2  Field Documentation . . . . .	78
6.32.2.1  id . . . . .	78
6.32.2.2  key . . . . .	78
6.32.2.3  action . . . . .	79
6.33  vtss_ece_tag_t Struct Reference . . . . .	79
6.33.1  Detailed Description . . . . .	79
6.33.2  Field Documentation . . . . .	79
6.33.2.1  vid . . . . .	79
6.33.2.2  pcp . . . . .	80
6.33.2.3  dei . . . . .	80
6.33.2.4  tagged . . . . .	80
6.33.2.5  s_tagged . . . . .	80

6.34 vtss_eee_port_conf_t Struct Reference . . . . .	80
6.34.1 Detailed Description . . . . .	81
6.34.2 Field Documentation . . . . .	81
6.34.2.1 eee_ena . . . . .	81
6.34.2.2 eee_fast_queues . . . . .	81
6.34.2.3 tx_tw . . . . .	81
6.34.2.4 lp_advertisement . . . . .	82
6.34.2.5 optimized_for_power . . . . .	82
6.35 vtss_eee_port_counter_t Struct Reference . . . . .	82
6.35.1 Detailed Description . . . . .	82
6.35.2 Field Documentation . . . . .	82
6.35.2.1 fill_level_get . . . . .	83
6.35.2.2 fill_level_thres . . . . .	83
6.35.2.3 fill_level . . . . .	83
6.35.2.4 tx_out_bytes_get . . . . .	83
6.35.2.5 tx_out_bytes . . . . .	83
6.36 vtss_eee_port_state_t Struct Reference . . . . .	84
6.36.1 Detailed Description . . . . .	84
6.36.2 Field Documentation . . . . .	84
6.36.2.1 select . . . . .	84
6.36.2.2 val . . . . .	84
6.37 vtss_eps_port_conf_t Struct Reference . . . . .	84
6.37.1 Detailed Description . . . . .	85
6.37.2 Field Documentation . . . . .	85
6.37.2.1 type . . . . .	85
6.37.2.2 port_no . . . . .	85
6.38 vtss_evc_conf_t Struct Reference . . . . .	85
6.38.1 Detailed Description . . . . .	86
6.38.2 Field Documentation . . . . .	86
6.38.2.1 learning . . . . .	86

6.38.2.2 pb . . . . .	86
6.38.2.3 network . . . . .	86
6.39 vtss_evc_inner_tag_t Struct Reference . . . . .	87
6.39.1 Detailed Description . . . . .	87
6.39.2 Field Documentation . . . . .	87
6.39.2.1 type . . . . .	87
6.39.2.2 vid_mode . . . . .	87
6.39.2.3 vid . . . . .	88
6.39.2.4 pcp_dei_preserve . . . . .	88
6.39.2.5 pcp . . . . .	88
6.39.2.6 dei . . . . .	88
6.40 vtss_evc_pb_conf_t Struct Reference . . . . .	88
6.40.1 Detailed Description . . . . .	89
6.40.2 Field Documentation . . . . .	89
6.40.2.1 nni . . . . .	89
6.40.2.2 ivid . . . . .	89
6.40.2.3 vid . . . . .	89
6.40.2.4 uvid . . . . .	90
6.40.2.5 inner_tag . . . . .	90
6.41 vtss_evc_port_conf_t Struct Reference . . . . .	90
6.41.1 Detailed Description . . . . .	90
6.41.2 Field Documentation . . . . .	90
6.41.2.1 dei_colouring . . . . .	91
6.41.2.2 inner_tag . . . . .	91
6.41.2.3 dmac_dip . . . . .	91
6.42 vtss_fan_conf_t Struct Reference . . . . .	91
6.42.1 Detailed Description . . . . .	92
6.42.2 Field Documentation . . . . .	92
6.42.2.1 fan_pwm_freq . . . . .	92
6.42.2.2 fan_low_pol . . . . .	92

6.42.2.3	fan_open_col . . . . .	92
6.42.2.4	type . . . . .	92
6.42.2.5	ppr . . . . .	93
6.43	vtss_init_conf_t Struct Reference . . . . .	93
6.43.1	Detailed Description . . . . .	93
6.43.2	Field Documentation . . . . .	93
6.43.2.1	reg_read . . . . .	94
6.43.2.2	reg_write . . . . .	94
6.43.2.3	miiim_read . . . . .	94
6.43.2.4	miiim_write . . . . .	94
6.43.2.5	mmd_read . . . . .	94
6.43.2.6	mmd_read_inc . . . . .	95
6.43.2.7	mmd_write . . . . .	95
6.43.2.8	spi_read_write . . . . .	95
6.43.2.9	spi_32bit_read_write . . . . .	95
6.43.2.10	spi_64bit_read_write . . . . .	95
6.43.2.11	warm_start_enable . . . . .	96
6.43.2.12	restart_info_src . . . . .	96
6.43.2.13	restart_info_port . . . . .	96
6.43.2.14	mux_mode . . . . .	96
6.43.2.15	pi . . . . .	96
6.43.2.16	serdes . . . . .	97
6.44	vtss_inst_create_t Struct Reference . . . . .	97
6.44.1	Detailed Description . . . . .	97
6.44.2	Field Documentation . . . . .	97
6.44.2.1	target . . . . .	97
6.45	vtss_intr_t Struct Reference . . . . .	98
6.45.1	Detailed Description . . . . .	98
6.45.2	Field Documentation . . . . .	98
6.45.2.1	link_change . . . . .	98

6.46 vtss_ip_addr_t Struct Reference . . . . .	98
6.46.1 Detailed Description . . . . .	99
6.46.2 Field Documentation . . . . .	99
6.46.2.1 type . . . . .	99
6.46.2.2 ipv4 . . . . .	99
6.46.2.3 ipv6 . . . . .	99
6.46.2.4 addr . . . . .	99
6.47 vtss_ip_network_t Struct Reference . . . . .	100
6.47.1 Detailed Description . . . . .	100
6.47.2 Field Documentation . . . . .	100
6.47.2.1 address . . . . .	100
6.47.2.2 prefix_size . . . . .	100
6.48 vtss_ipv4_network_t Struct Reference . . . . .	100
6.48.1 Detailed Description . . . . .	101
6.48.2 Field Documentation . . . . .	101
6.48.2.1 address . . . . .	101
6.48.2.2 prefix_size . . . . .	101
6.49 vtss_ipv4_uc_t Struct Reference . . . . .	101
6.49.1 Detailed Description . . . . .	102
6.49.2 Field Documentation . . . . .	102
6.49.2.1 network . . . . .	102
6.49.2.2 destination . . . . .	102
6.50 vtss_ipv6_network_t Struct Reference . . . . .	102
6.50.1 Detailed Description . . . . .	103
6.50.2 Field Documentation . . . . .	103
6.50.2.1 address . . . . .	103
6.50.2.2 prefix_size . . . . .	103
6.51 vtss_ipv6_t Struct Reference . . . . .	103
6.51.1 Detailed Description . . . . .	103
6.51.2 Field Documentation . . . . .	104

---

6.51.2.1	addr	104
6.52	vtss_ipv6_uc_t Struct Reference	104
6.52.1	Detailed Description	104
6.52.2	Field Documentation	104
6.52.2.1	network	104
6.52.2.2	destination	105
6.53	vtss_irq_conf_t Struct Reference	105
6.53.1	Detailed Description	105
6.53.2	Field Documentation	105
6.53.2.1	external	105
6.53.2.2	destination	106
6.54	vtss_irq_status_t Struct Reference	106
6.54.1	Detailed Description	106
6.54.2	Field Documentation	106
6.54.2.1	active	106
6.54.2.2	raw_ident	107
6.54.2.3	raw_status	107
6.54.2.4	raw_mask	107
6.55	vtss_l3_counters_t Struct Reference	107
6.55.1	Detailed Description	108
6.55.2	Field Documentation	108
6.55.2.1	ipv4uc_received_octets	108
6.55.2.2	ipv4uc_received_frames	108
6.55.2.3	ipv6uc_received_octets	108
6.55.2.4	ipv6uc_received_frames	108
6.55.2.5	ipv4uc_transmitted_octets	109
6.55.2.6	ipv4uc_transmitted_frames	109
6.55.2.7	ipv6uc_transmitted_octets	109
6.55.2.8	ipv6uc_transmitted_frames	109
6.56	vtss_lcpll_status_t Struct Reference	109

6.56.1	Detailed Description	110
6.56.2	Field Documentation	110
6.56.2.1	lock_status	110
6.56.2.2	cal_done	110
6.56.2.3	cal_error	110
6.56.2.4	fsm_lock	111
6.56.2.5	fsm_stat	111
6.56.2.6	gain_stat	111
6.57	vtss_learn_mode_t Struct Reference	111
6.57.1	Detailed Description	111
6.57.2	Field Documentation	112
6.57.2.1	automatic	112
6.57.2.2	cpu	112
6.57.2.3	discard	112
6.58	vtss_mac_t Struct Reference	112
6.58.1	Detailed Description	113
6.58.2	Field Documentation	113
6.58.2.1	addr	113
6.59	vtss_mac_table_entry_t Struct Reference	113
6.59.1	Detailed Description	113
6.59.2	Field Documentation	113
6.59.2.1	vid_mac	114
6.59.2.2	destination	114
6.59.2.3	copy_to_cpu	114
6.59.2.4	locked	114
6.59.2.5	aged	114
6.59.2.6	cpu_queue	115
6.60	vtss_mac_table_status_t Struct Reference	115
6.60.1	Detailed Description	115
6.60.2	Field Documentation	115

---

6.60.2.1 learned . . . . .	115
6.60.2.2 replaced . . . . .	116
6.60.2.3 moved . . . . .	116
6.60.2.4 aged . . . . .	116
6.61 vtss_mce_action_t Struct Reference . . . . .	116
6.61.1 Detailed Description . . . . .	117
6.61.2 Field Documentation . . . . .	117
6.61.2.1 policy_no . . . . .	117
6.61.2.2 prio_enable . . . . .	117
6.61.2.3 prio . . . . .	117
6.61.2.4 vid . . . . .	117
6.61.2.5 pop_cnt . . . . .	118
6.62 vtss_mce_key_t Struct Reference . . . . .	118
6.62.1 Detailed Description . . . . .	118
6.62.2 Field Documentation . . . . .	118
6.62.2.1 port_list . . . . .	118
6.62.2.2 vid . . . . .	119
6.62.2.3 data . . . . .	119
6.63 vtss_mce_t Struct Reference . . . . .	119
6.63.1 Detailed Description . . . . .	119
6.63.2 Field Documentation . . . . .	119
6.63.2.1 tt_loop . . . . .	120
6.63.2.2 id . . . . .	120
6.63.2.3 key . . . . .	120
6.63.2.4 action . . . . .	120
6.64 vtss_mirror_conf_t Struct Reference . . . . .	120
6.64.1 Detailed Description . . . . .	121
6.64.2 Field Documentation . . . . .	121
6.64.2.1 port_no . . . . .	121
6.64.2.2 fwd_enable . . . . .	121

6.65 <code>vtss_mtimer_t</code> Struct Reference . . . . .	121
6.65.1 Detailed Description . . . . .	122
6.65.2 Field Documentation . . . . .	122
6.65.2.1 <code>timeout</code> . . . . .	122
6.65.2.2 <code>now</code> . . . . .	122
6.66 <code>vtss_npi_conf_t</code> Struct Reference . . . . .	122
6.66.1 Detailed Description . . . . .	123
6.66.2 Field Documentation . . . . .	123
6.66.2.1 <code>enable</code> . . . . .	123
6.66.2.2 <code>port_no</code> . . . . .	123
6.67 <code>vtss_os_timestamp_t</code> Struct Reference . . . . .	123
6.67.1 Detailed Description . . . . .	123
6.67.2 Field Documentation . . . . .	124
6.67.2.1 <code>hw_cnt</code> . . . . .	124
6.68 <code>vtss_packet_dma_conf_t</code> Struct Reference . . . . .	124
6.68.1 Detailed Description . . . . .	124
6.68.2 Field Documentation . . . . .	124
6.68.2.1 <code>dma_enable</code> . . . . .	124
6.69 <code>vtss_packet_frame_info_t</code> Struct Reference . . . . .	125
6.69.1 Detailed Description . . . . .	125
6.69.2 Field Documentation . . . . .	125
6.69.2.1 <code>port_no</code> . . . . .	125
6.69.2.2 <code>vid</code> . . . . .	125
6.69.2.3 <code>port_tx</code> . . . . .	126
6.69.2.4 <code>aggr_rx_disable</code> . . . . .	126
6.69.2.5 <code>aggr_tx_disable</code> . . . . .	126
6.70 <code>vtss_packet_port_filter_t</code> Struct Reference . . . . .	126
6.70.1 Detailed Description . . . . .	126
6.70.2 Field Documentation . . . . .	127
6.70.2.1 <code>filter</code> . . . . .	127

6.70.2.2 <code>tpid</code>	127
6.71 <code>vtss_packet_port_info_t</code> Struct Reference	127
6.71.1 Detailed Description	127
6.71.2 Field Documentation	128
6.71.2.1 <code>port_no</code>	128
6.71.2.2 <code>vid</code>	128
6.71.2.3 <code>aggr_rx_disable</code>	128
6.71.2.4 <code>aggr_tx_disable</code>	128
6.72 <code>vtss_packet_rx_conf_t</code> Struct Reference	129
6.72.1 Detailed Description	129
6.72.2 Field Documentation	129
6.72.2.1 <code>queue</code>	129
6.72.2.2 <code>reg</code>	129
6.72.2.3 <code>map</code>	130
6.72.2.4 <code>grp_map</code>	130
6.73 <code>vtss_packet_rx_header_t</code> Struct Reference	130
6.73.1 Detailed Description	130
6.73.2 Field Documentation	130
6.73.2.1 <code>length</code>	131
6.73.2.2 <code>port_no</code>	131
6.73.2.3 <code>queue_mask</code>	131
6.73.2.4 <code>learn</code>	131
6.73.2.5 <code>arrived_tagged</code>	131
6.73.2.6 <code>tag</code>	132
6.74 <code>vtss_packet_rx_info_t</code> Struct Reference	132
6.74.1 Detailed Description	132
6.74.2 Field Documentation	133
6.74.2.1 <code>hints</code>	133
6.74.2.2 <code>length</code>	133
6.74.2.3 <code>port_no</code>	134

6.74.2.4	glag_no	134
6.74.2.5	tag_type	135
6.74.2.6	tag	135
6.74.2.7	stripped_tag	136
6.74.2.8	xtr_qu_mask	136
6.74.2.9	cos	137
6.74.2.10	acl_hit	137
6.74.2.11	acl_idx	137
6.74.2.12	sw_tstamp	138
6.74.2.13	tstamp_id	138
6.74.2.14	tstamp_id_decoded	138
6.74.2.15	hw_tstamp	139
6.74.2.16	hw_tstamp_decoded	139
6.74.2.17	sflow_type	139
6.74.2.18	sflow_port_no	140
6.74.2.19	oam_info	140
6.74.2.20	oam_info_decoded	140
6.74.2.21	isdx	141
6.75	vtss_packet_rx_meta_t Struct Reference	141
6.75.1	Detailed Description	141
6.75.2	Field Documentation	142
6.75.2.1	no_wait	142
6.75.2.2	chip_no	142
6.75.2.3	xtr_qu	143
6.75.2.4	etype	143
6.75.2.5	fcs	144
6.75.2.6	sw_tstamp	144
6.75.2.7	length	145
6.76	vtss_packet_rx_port_conf_t Struct Reference	145
6.76.1	Detailed Description	145

---

6.76.2 Field Documentation . . . . .	146
6.76.2.1 bpdu_reg . . . . .	146
6.76.2.2 garp_reg . . . . .	146
6.77 vtss_packet_rx_queue_conf_t Struct Reference . . . . .	146
6.77.1 Detailed Description . . . . .	146
6.77.2 Field Documentation . . . . .	146
6.77.2.1 size . . . . .	147
6.77.2.2 npi . . . . .	147
6.78 vtss_packet_rx_queue_map_t Struct Reference . . . . .	147
6.78.1 Detailed Description . . . . .	147
6.78.2 Field Documentation . . . . .	148
6.78.2.1 bpdu_queue . . . . .	148
6.78.2.2 garp_queue . . . . .	148
6.78.2.3 learn_queue . . . . .	148
6.78.2.4 igmp_queue . . . . .	148
6.78.2.5 ipmc_ctrl_queue . . . . .	148
6.78.2.6 mac_vid_queue . . . . .	149
6.78.2.7 stack_queue . . . . .	149
6.78.2.8 sflow_queue . . . . .	149
6.78.2.9 lrn_all_queue . . . . .	149
6.79 vtss_packet_rx_queue_npi_conf_t Struct Reference . . . . .	149
6.79.1 Detailed Description . . . . .	150
6.79.2 Field Documentation . . . . .	150
6.79.2.1 enable . . . . .	150
6.80 vtss_packet_rx_reg_t Struct Reference . . . . .	150
6.80.1 Detailed Description . . . . .	150
6.80.2 Field Documentation . . . . .	151
6.80.2.1 bpdu_cpu_only . . . . .	151
6.80.2.2 garp_cpu_only . . . . .	151
6.80.2.3 ipmc_ctrl_cpu_copy . . . . .	151

6.80.2.4 igmp_cpu_only . . . . .	151
6.80.2.5 mld_cpu_only . . . . .	152
6.81 vtss_packet_tx_ifh_t Struct Reference . . . . .	152
6.81.1 Detailed Description . . . . .	152
6.81.2 Field Documentation . . . . .	152
6.81.2.1 length . . . . .	152
6.81.2.2 ifh . . . . .	153
6.82 vtss_packet_tx_info_t Struct Reference . . . . .	153
6.82.1 Detailed Description . . . . .	153
6.82.2 Field Documentation . . . . .	154
6.82.2.1 switch_frm . . . . .	154
6.82.2.2 dst_port_mask . . . . .	154
6.82.2.3 frm_len . . . . .	155
6.82.2.4 tag . . . . .	155
6.82.2.5 aggr_code . . . . .	156
6.82.2.6 cos . . . . .	157
6.82.2.7 ptp_action . . . . .	157
6.82.2.8 ptp_id . . . . .	158
6.82.2.9 ptp_timestamp . . . . .	158
6.82.2.10 latch_timestamp . . . . .	159
6.82.2.11 oam_type . . . . .	159
6.82.2.12 isdx . . . . .	160
6.82.2.13 isdx_dont_use . . . . .	160
6.82.2.14 dp . . . . .	161
6.82.2.15 masquerade_port . . . . .	161
6.82.2.16 pdu_offset . . . . .	162
6.83 vtss_phy_aneg_t Struct Reference . . . . .	162
6.83.1 Detailed Description . . . . .	162
6.83.2 Field Documentation . . . . .	163
6.83.2.1 speed_10m_hdx . . . . .	163

6.83.2.2	speed_10m_fdx	163
6.83.2.3	speed_100m_hdx	163
6.83.2.4	speed_100m_fdx	163
6.83.2.5	speed_1g_fdx	163
6.83.2.6	speed_1g_hdx	164
6.83.2.7	symmetric_pause	164
6.83.2.8	asymmetric_pause	164
6.83.2.9	tx_remote_fault	164
6.84	vtss_phy_clock_conf_t Struct Reference	164
6.84.1	Detailed Description	165
6.84.2	Field Documentation	165
6.84.2.1	src	165
6.84.2.2	freq	165
6.84.2.3	squelch	165
6.85	vtss_phy_conf_1g_t Struct Reference	166
6.85.1	Detailed Description	166
6.85.2	Field Documentation	166
6.85.2.1	cfg	166
6.85.2.2	val	166
6.85.2.3	master	167
6.86	vtss_phy_conf_t Struct Reference	167
6.86.1	Detailed Description	167
6.86.2	Field Documentation	167
6.86.2.1	mode	167
6.86.2.2	forced	168
6.86.2.3	aneg	168
6.86.2.4	mdi	168
6.86.2.5	fif	168
6.86.2.6	sigdet	168
6.86.2.7	unidir	169

6.86.2.8 mac_if_pcs . . . . .	169
6.86.2.9 media_if_pcs . . . . .	169
6.86.2.10 force_ams_sel . . . . .	169
6.86.2.11 skip_coma . . . . .	169
6.87 vtss_phy_eee_conf_t Struct Reference . . . . .	170
6.87.1 Detailed Description . . . . .	170
6.87.2 Field Documentation . . . . .	170
6.87.2.1 eee_mode . . . . .	170
6.87.2.2 eee_ena_phy . . . . .	170
6.88 vtss_phy_enhanced_led_control_t Struct Reference . . . . .	170
6.88.1 Detailed Description . . . . .	171
6.88.2 Field Documentation . . . . .	171
6.88.2.1 ser_led_output_1 . . . . .	171
6.88.2.2 ser_led_output_2 . . . . .	171
6.88.2.3 ser_led_frame_rate . . . . .	171
6.88.2.4 ser_led_select . . . . .	172
6.89 vtss_phy_forced_t Struct Reference . . . . .	172
6.89.1 Detailed Description . . . . .	172
6.89.2 Field Documentation . . . . .	172
6.89.2.1 speed . . . . .	172
6.89.2.2 fdx . . . . .	173
6.90 vtss_phy_led_mode_select_t Struct Reference . . . . .	173
6.90.1 Detailed Description . . . . .	173
6.90.2 Field Documentation . . . . .	173
6.90.2.1 mode . . . . .	173
6.90.2.2 number . . . . .	174
6.91 vtss_phy_loopback_t Struct Reference . . . . .	174
6.91.1 Detailed Description . . . . .	174
6.91.2 Field Documentation . . . . .	174
6.91.2.1 far_end_enable . . . . .	174

6.91.2.2	near_end_enable	175
6.91.2.3	connector_enable	175
6.91.2.4	mac_serdes_input_enable	175
6.91.2.5	mac_serdes_facility_enable	175
6.91.2.6	mac_serdes_equipment_enable	175
6.91.2.7	media_serdes_input_enable	176
6.91.2.8	media_serdes_facility_enable	176
6.91.2.9	media_serdes_equipment_enable	176
6.92	vtss_phy_mac_serd_pcs_cntl_t Struct Reference	176
6.92.1	Detailed Description	177
6.92.2	Field Documentation	177
6.92.2.1	disable	177
6.92.2.2	restart	177
6.92.2.3	pd_enable	177
6.92.2.4	aneg_restart	177
6.92.2.5	force_adv_ability	178
6.92.2.6	sgmii_in_pre	178
6.92.2.7	sgmii_out_pre	178
6.92.2.8	serdes_aneg_ena	178
6.92.2.9	serdes_pol_inv_in	178
6.92.2.10	serdes_pol_inv_out	179
6.92.2.11	fast_link_stat_ena	179
6.92.2.12	inhibit_odd_start	179
6.93	vtss_phy_media_serd_pcs_cntl_t Struct Reference	179
6.93.1	Detailed Description	180
6.93.2	Field Documentation	180
6.93.2.1	remote_fault	180
6.93.2.2	aneg_pd_detect	180
6.93.2.3	force_adv_ability	180
6.93.2.4	serdes_pol_inv_in	180

6.93.2.5  serdes_pol_inv_out . . . . .	181
6.93.2.6  inhibit_odd_start . . . . .	181
6.93.2.7  force_hls . . . . .	181
6.93.2.8  force_fefi . . . . .	181
6.93.2.9  force_fefi_value . . . . .	181
6.94  vtss_phy_power_conf_t Struct Reference . . . . .	182
6.94.1  Detailed Description . . . . .	182
6.94.2  Field Documentation . . . . .	182
6.94.2.1  mode . . . . .	182
6.95  vtss_phy_power_status_t Struct Reference . . . . .	182
6.95.1  Detailed Description . . . . .	183
6.95.2  Field Documentation . . . . .	183
6.95.2.1  level . . . . .	183
6.96  vtss_phy_reset_conf_t Struct Reference . . . . .	183
6.96.1  Detailed Description . . . . .	183
6.96.2  Field Documentation . . . . .	184
6.96.2.1  mac_if . . . . .	184
6.96.2.2  media_if . . . . .	184
6.96.2.3  rgmii . . . . .	184
6.96.2.4  tbi . . . . .	184
6.96.2.5  force . . . . .	184
6.96.2.6  pkt_mode . . . . .	185
6.96.2.7  i_cpu_en . . . . .	185
6.97  vtss_phy_rgmii_conf_t Struct Reference . . . . .	185
6.97.1  Detailed Description . . . . .	185
6.97.2  Field Documentation . . . . .	185
6.97.2.1  rx_clk_skew_ps . . . . .	186
6.97.2.2  tx_clk_skew_ps . . . . .	186
6.98  vtss_phy_statistic_t Struct Reference . . . . .	186
6.98.1  Detailed Description . . . . .	186

---

6.98.2 Field Documentation . . . . .	186
6.98.2.1 cu_good . . . . .	187
6.98.2.2 cu_bad . . . . .	187
6.98.2.3 serdes_tx_good . . . . .	187
6.98.2.4 serdes_tx_bad . . . . .	187
6.98.2.5 rx_err_cnt_base_tx . . . . .	187
6.98.2.6 media_mac_serdes_good . . . . .	188
6.98.2.7 media_mac_serdes_crc . . . . .	188
6.99 vtss_phy_status_1g_t Struct Reference . . . . .	188
6.99.1 Detailed Description . . . . .	188
6.99.2 Field Documentation . . . . .	188
6.99.2.1 master_cfg_fault . . . . .	189
6.99.2.2 master . . . . .	189
6.100vtss_phy_tbi_conf_t Struct Reference . . . . .	189
6.100.1 Detailed Description . . . . .	189
6.100.2 Field Documentation . . . . .	189
6.100.2.1 aneg_enable . . . . .	190
6.101vtss_phy_type_t Struct Reference . . . . .	190
6.101.1 Detailed Description . . . . .	190
6.101.2 Field Documentation . . . . .	190
6.101.2.1 part_number . . . . .	190
6.101.2.2 revision . . . . .	191
6.101.2.3 port_cnt . . . . .	191
6.101.2.4 channel_id . . . . .	191
6.101.2.5 base_port_no . . . . .	191
6.101.2.6 phy_api_base_no . . . . .	191
6.102vtss_phy_veriphy_result_t Struct Reference . . . . .	192
6.102.1 Detailed Description . . . . .	192
6.102.2 Field Documentation . . . . .	192
6.102.2.1 link . . . . .	192

6.102.2.2 status . . . . .	192
6.102.2.3 length . . . . .	193
6.103vtss_phy_wol_conf_t Struct Reference . . . . .	193
6.103.1 Detailed Description . . . . .	193
6.103.2 Field Documentation . . . . .	193
6.103.2.1 secure_on_enable . . . . .	193
6.103.2.2 wol_mac . . . . .	194
6.103.2.3 wol_pass . . . . .	194
6.103.2.4 wol_passwd_len . . . . .	194
6.103.2.5 magic_pkt_cnt . . . . .	194
6.104vtss_pi_conf_t Struct Reference . . . . .	194
6.104.1 Detailed Description . . . . .	195
6.104.2 Field Documentation . . . . .	195
6.104.2.1 width . . . . .	195
6.104.2.2 use_extended_bus_cycle . . . . .	195
6.104.2.3 cs_wait_ns . . . . .	195
6.105vtss_policer_ext_t Struct Reference . . . . .	196
6.105.1 Detailed Description . . . . .	196
6.105.2 Field Documentation . . . . .	196
6.105.2.1 frame_rate . . . . .	196
6.105.2.2 flow_control . . . . .	196
6.106vtss_policer_t Struct Reference . . . . .	196
6.106.1 Detailed Description . . . . .	197
6.106.2 Field Documentation . . . . .	197
6.106.2.1 level . . . . .	197
6.106.2.2 rate . . . . .	197
6.107vtss_port_bridge_counters_t Struct Reference . . . . .	197
6.107.1 Detailed Description . . . . .	198
6.107.2 Field Documentation . . . . .	198
6.107.2.1 dot1dTpPortInDiscards . . . . .	198

6.108vtss_port_clause_37_adv_t Struct Reference . . . . .	198
6.108.1 Detailed Description . . . . .	198
6.108.2 Field Documentation . . . . .	199
6.108.2.1 fdx . . . . .	199
6.108.2.2 hdx . . . . .	199
6.108.2.3 symmetric_pause . . . . .	199
6.108.2.4 asymmetric_pause . . . . .	199
6.108.2.5 remote_fault . . . . .	199
6.108.2.6 acknowledge . . . . .	200
6.108.2.7 next_page . . . . .	200
6.109vtss_port_clause_37_control_t Struct Reference . . . . .	200
6.109.1 Detailed Description . . . . .	200
6.109.2 Field Documentation . . . . .	200
6.109.2.1 enable . . . . .	201
6.109.2.2 advertisement . . . . .	201
6.110vtss_port_conf_t Struct Reference . . . . .	201
6.110.1 Detailed Description . . . . .	202
6.110.2 Field Documentation . . . . .	202
6.110.2.1 if_type . . . . .	202
6.110.2.2 sd_enable . . . . .	202
6.110.2.3 sd_active_high . . . . .	202
6.110.2.4 sd_internal . . . . .	202
6.110.2.5 frame_gaps . . . . .	203
6.110.2.6 power_down . . . . .	203
6.110.2.7 speed . . . . .	203
6.110.2.8 fdx . . . . .	203
6.110.2.9 flow_control . . . . .	203
6.110.2.10max_frame_length . . . . .	204
6.110.2.11frame_length_chk . . . . .	204
6.110.2.12max_tags . . . . .	204

6.110.2.13exc_col_cont . . . . .	204
6.110.2.14xaui_rx_lane_flip . . . . .	204
6.110.2.15xaui_tx_lane_flip . . . . .	205
6.110.2.16oop . . . . .	205
6.110.2.17serdes . . . . .	205
<b>6.111vtss_port_counters_t Struct Reference</b> . . . . .	205
<b>6.111.1 Detailed Description</b> . . . . .	206
<b>6.111.2 Field Documentation</b> . . . . .	206
6.111.2.1 rmon . . . . .	206
6.111.2.2 if_group . . . . .	206
6.111.2.3 ethernet_like . . . . .	206
6.111.2.4 bridge . . . . .	206
6.111.2.5 prop . . . . .	207
6.111.2.6 evc . . . . .	207
<b>6.112vtss_port_ethernet_like_counters_t Struct Reference</b> . . . . .	207
<b>6.112.1 Detailed Description</b> . . . . .	207
<b>6.112.2 Field Documentation</b> . . . . .	207
6.112.2.1 dot3InPauseFrames . . . . .	208
6.112.2.2 dot3OutPauseFrames . . . . .	208
<b>6.113vtss_port_evc_counters_t Struct Reference</b> . . . . .	208
<b>6.113.1 Detailed Description</b> . . . . .	208
<b>6.113.2 Field Documentation</b> . . . . .	208
6.113.2.1 rx_green . . . . .	209
6.113.2.2 rx_yellow . . . . .	209
6.113.2.3 rx_red . . . . .	209
6.113.2.4 rx_green_discard . . . . .	209
6.113.2.5 rx_yellow_discard . . . . .	209
6.113.2.6 tx_green . . . . .	210
6.113.2.7 tx_yellow . . . . .	210
<b>6.114vtss_port_flow_control_conf_t Struct Reference</b> . . . . .	210

6.114.1 Detailed Description . . . . .	210
6.114.2 Field Documentation . . . . .	210
6.114.2.1 obey . . . . .	211
6.114.2.2 generate . . . . .	211
6.114.2.3 smac . . . . .	211
6.114.2.4 pfc . . . . .	211
6.115vtss_port_frame_gaps_t Struct Reference . . . . .	211
6.115.1 Detailed Description . . . . .	212
6.115.2 Field Documentation . . . . .	212
6.115.2.1 hdx_gap_1 . . . . .	212
6.115.2.2 hdx_gap_2 . . . . .	212
6.115.2.3 fdx_gap . . . . .	212
6.116vtss_port_if_group_counters_t Struct Reference . . . . .	213
6.116.1 Detailed Description . . . . .	213
6.116.2 Field Documentation . . . . .	213
6.116.2.1 ifInOctets . . . . .	213
6.116.2.2 ifInUcastPkts . . . . .	213
6.116.2.3 ifInMulticastPkts . . . . .	214
6.116.2.4 ifInBroadcastPkts . . . . .	214
6.116.2.5 ifInNUcastPkts . . . . .	214
6.116.2.6 ifInDiscards . . . . .	214
6.116.2.7 ifInErrors . . . . .	214
6.116.2.8 ifOutOctets . . . . .	215
6.116.2.9 ifOutUcastPkts . . . . .	215
6.116.2.10fOutMulticastPkts . . . . .	215
6.116.2.11fOutBroadcastPkts . . . . .	215
6.116.2.12fOutNUcastPkts . . . . .	215
6.116.2.13fOutDiscards . . . . .	216
6.116.2.14fOutErrors . . . . .	216
6.117vtss_port_map_t Struct Reference . . . . .	216

6.117.1 Detailed Description . . . . .	216
6.117.2 Field Documentation . . . . .	216
6.117.2.1 chip_port . . . . .	217
6.117.2.2 chip_no . . . . .	217
6.117.2.3 miim_controller . . . . .	217
6.117.2.4 miim_addr . . . . .	217
6.117.2.5 miim_chip_no . . . . .	217
6.118vtss_port_proprietary_counters_t Struct Reference . . . . .	218
6.118.1 Detailed Description . . . . .	218
6.118.2 Field Documentation . . . . .	218
6.118.2.1 rx_prio . . . . .	218
6.118.2.2 tx_prio . . . . .	218
6.119vtss_port_rmon_counters_t Struct Reference . . . . .	218
6.119.1 Detailed Description . . . . .	219
6.119.2 Field Documentation . . . . .	219
6.119.2.1 rx_etherStatsDropEvents . . . . .	219
6.119.2.2 rx_etherStatsOctets . . . . .	220
6.119.2.3 rx_etherStatsPkts . . . . .	220
6.119.2.4 rx_etherStatsBroadcastPkts . . . . .	220
6.119.2.5 rx_etherStatsMulticastPkts . . . . .	220
6.119.2.6 rx_etherStatsCRCAlignErrors . . . . .	220
6.119.2.7 rx_etherStatsUndersizePkts . . . . .	221
6.119.2.8 rx_etherStatsOversizePkts . . . . .	221
6.119.2.9 rx_etherStatsFragments . . . . .	221
6.119.2.10rx_etherStatsJabbers . . . . .	221
6.119.2.11rx_etherStatsPkts64Octets . . . . .	221
6.119.2.12rx_etherStatsPkts65to127Octets . . . . .	222
6.119.2.13rx_etherStatsPkts128to255Octets . . . . .	222
6.119.2.14rx_etherStatsPkts256to511Octets . . . . .	222
6.119.2.15rx_etherStatsPkts512to1023Octets . . . . .	222

6.119.2.10x_etherStatsPkts1024to1518Octets . . . . .	222
6.119.2.17x_etherStatsPkts1519toMaxOctets . . . . .	223
6.119.2.18x_etherStatsDropEvents . . . . .	223
6.119.2.19x_etherStatsOctets . . . . .	223
6.119.2.20x_etherStatsPkts . . . . .	223
6.119.2.21tx_etherStatsBroadcastPkts . . . . .	223
6.119.2.22tx_etherStatsMulticastPkts . . . . .	224
6.119.2.23tx_etherStatsCollisions . . . . .	224
6.119.2.24tx_etherStatsPkts64Octets . . . . .	224
6.119.2.25tx_etherStatsPkts65to127Octets . . . . .	224
6.119.2.26tx_etherStatsPkts128to255Octets . . . . .	224
6.119.2.27tx_etherStatsPkts256to511Octets . . . . .	225
6.119.2.28tx_etherStatsPkts512to1023Octets . . . . .	225
6.119.2.29tx_etherStatsPkts1024to1518Octets . . . . .	225
6.119.2.30tx_etherStatsPkts1519toMaxOctets . . . . .	225
6.120vtss_port_serdes_conf_t Struct Reference . . . . .	225
6.120.1 Detailed Description . . . . .	226
6.120.2 Field Documentation . . . . .	226
6.120.2.1 sfp_dac . . . . .	226
6.121vtss_port_sgmii_aneg_t Struct Reference . . . . .	226
6.121.1 Detailed Description . . . . .	226
6.121.2 Field Documentation . . . . .	227
6.121.2.1 link . . . . .	227
6.121.2.2 fdx . . . . .	227
6.121.2.3 hdx . . . . .	227
6.121.2.4 speed_10M . . . . .	227
6.121.2.5 speed_100M . . . . .	227
6.121.2.6 speed_1G . . . . .	228
6.121.2.7 aneg_complete . . . . .	228
6.122vtss_port_status_t Struct Reference . . . . .	228

6.122.1 Detailed Description . . . . .	228
6.122.2 Field Documentation . . . . .	229
6.122.2.1 link_down . . . . .	229
6.122.2.2 link . . . . .	229
6.122.2.3 speed . . . . .	229
6.122.2.4 fdx . . . . .	229
6.122.2.5 remote_fault . . . . .	229
6.122.2.6 aneg_complete . . . . .	230
6.122.2.7 unidirectional_ability . . . . .	230
6.122.2.8 aneg . . . . .	230
6.122.2.9 mdi_cross . . . . .	230
6.122.2.10 fiber . . . . .	230
6.122.2.11 copper . . . . .	231
6.123vtss_qce_action_t Struct Reference . . . . .	231
6.123.1 Detailed Description . . . . .	231
6.123.2 Field Documentation . . . . .	231
6.123.2.1 prio_enable . . . . .	231
6.123.2.2 prio . . . . .	232
6.123.2.3 dp_enable . . . . .	232
6.123.2.4 dp . . . . .	232
6.123.2.5 dscp_enable . . . . .	232
6.123.2.6 dscp . . . . .	232
6.123.2.7 pcp_dei_enable . . . . .	233
6.123.2.8 pcp . . . . .	233
6.123.2.9 dei . . . . .	233
6.123.2.10 policy_no_enable . . . . .	233
6.123.2.11 policy_no . . . . .	233
6.124vtss_qce_frame_etype_t Struct Reference . . . . .	234
6.124.1 Detailed Description . . . . .	234
6.124.2 Field Documentation . . . . .	234

6.124.2.1 <code>etype</code>	234
6.124.2.2 <code>data</code>	234
6.125vtss_qce_frame_ipv4_t Struct Reference	234
6.125.1 Detailed Description	235
6.125.2 Field Documentation	235
6.125.2.1 <code>fragment</code>	235
6.125.2.2 <code>dscp</code>	235
6.125.2.3 <code>proto</code>	235
6.125.2.4 <code>sip</code>	236
6.125.2.5 <code>sport</code>	236
6.125.2.6 <code>dport</code>	236
6.126vtss_qce_frame_ipv6_t Struct Reference	236
6.126.1 Detailed Description	237
6.126.2 Field Documentation	237
6.126.2.1 <code>dscp</code>	237
6.126.2.2 <code>proto</code>	237
6.126.2.3 <code>sip</code>	237
6.126.2.4 <code>sport</code>	237
6.126.2.5 <code>dport</code>	238
6.127vtss_qce_frame_llc_t Struct Reference	238
6.127.1 Detailed Description	238
6.127.2 Field Documentation	238
6.127.2.1 <code>data</code>	238
6.128vtss_qce_frame_snap_t Struct Reference	239
6.128.1 Detailed Description	239
6.128.2 Field Documentation	239
6.128.2.1 <code>data</code>	239
6.129vtss_qce_key_t Struct Reference	239
6.129.1 Detailed Description	240
6.129.2 Field Documentation	240

---

6.129.2.1 port_list . . . . .	240
6.129.2.2 mac . . . . .	240
6.129.2.3 tag . . . . .	240
6.129.2.4 type . . . . .	240
6.129.2.5 etype . . . . .	241
6.129.2.6 llc . . . . .	241
6.129.2.7 snap . . . . .	241
6.129.2.8 ipv4 . . . . .	241
6.129.2.9 ipv6 . . . . .	241
6.129.2.10 frame . . . . .	242
6.130 vtss_qce_mac_t Struct Reference . . . . .	242
6.130.1 Detailed Description . . . . .	242
6.130.2 Field Documentation . . . . .	242
6.130.2.1 dmac_mc . . . . .	242
6.130.2.2 dmac_bc . . . . .	243
6.130.2.3 smac . . . . .	243
6.131 vtss_qce_t Struct Reference . . . . .	243
6.131.1 Detailed Description . . . . .	243
6.131.2 Field Documentation . . . . .	243
6.131.2.1 id . . . . .	244
6.131.2.2 key . . . . .	244
6.131.2.3 action . . . . .	244
6.132 vtss_qce_tag_t Struct Reference . . . . .	244
6.132.1 Detailed Description . . . . .	245
6.132.2 Field Documentation . . . . .	245
6.132.2.1 vid . . . . .	245
6.132.2.2 pcp . . . . .	245
6.132.2.3 dei . . . . .	245
6.132.2.4 tagged . . . . .	245
6.132.2.5 s_tag . . . . .	246

6.133vtss_qos_conf_t Struct Reference . . . . .	246
6.133.1 Detailed Description . . . . .	246
6.133.2 Field Documentation . . . . .	246
6.133.2.1 prios . . . . .	247
6.133.2.2 dscp_trust . . . . .	247
6.133.2.3 dscp_qos_class_map . . . . .	247
6.133.2.4 dscp_dp_level_map . . . . .	247
6.133.2.5 dscp_qos_map . . . . .	247
6.133.2.6 dscp_qos_map_dp1 . . . . .	248
6.133.2.7 dscp_remark . . . . .	248
6.133.2.8 dscp_translate_map . . . . .	248
6.133.2.9 dscp_remap . . . . .	248
6.133.2.10dscp_remap_dp1 . . . . .	248
6.133.2.11policer_uc . . . . .	249
6.133.2.12policer_uc_frame_rate . . . . .	249
6.133.2.13policer_uc_mode . . . . .	249
6.133.2.14policer_mc . . . . .	249
6.133.2.15policer_mc_frame_rate . . . . .	249
6.133.2.16policer_mc_mode . . . . .	250
6.133.2.17policer_bc . . . . .	250
6.133.2.18policer_bc_frame_rate . . . . .	250
6.133.2.19policer_bc_mode . . . . .	250
6.134vtss_qos_port_conf_t Struct Reference . . . . .	250
6.134.1 Detailed Description . . . . .	251
6.134.2 Field Documentation . . . . .	251
6.134.2.1 policer_port . . . . .	251
6.134.2.2 policer_ext_port . . . . .	252
6.134.2.3 policer_queue . . . . .	252
6.134.2.4 shaper_port . . . . .	252
6.134.2.5 shaper_queue . . . . .	252

6.134.2.6 excess_enable . . . . .	252
6.134.2.7 default_prio . . . . .	253
6.134.2.8 usr_prio . . . . .	253
6.134.2.9 default_dpl . . . . .	253
6.134.2.10 default_dei . . . . .	253
6.134.2.11 tag_class_enable . . . . .	253
6.134.2.12 qos_class_map . . . . .	254
6.134.2.13 dp_level_map . . . . .	254
6.134.2.14 dscp_class_enable . . . . .	254
6.134.2.15 dscp_mode . . . . .	254
6.134.2.16 dscp_emode . . . . .	254
6.134.2.17 dscp_translate . . . . .	255
6.134.2.18 ag_remark_mode . . . . .	255
6.134.2.19 ag_default_pcp . . . . .	255
6.134.2.20 tag_default_dei . . . . .	255
6.134.2.21 tag_pcp_map . . . . .	255
6.134.2.22 tag_dei_map . . . . .	256
6.134.2.23 dwrr_enable . . . . .	256
6.134.2.24 queue_pct . . . . .	256
6.134.2.25 dmac_dip . . . . .	256
6.135 vtss_rcpll_status_t Struct Reference . . . . .	256
6.135.1 Detailed Description . . . . .	257
6.135.2 Field Documentation . . . . .	257
6.135.2.1 out_of_range . . . . .	257
6.135.2.2 cal_error . . . . .	257
6.135.2.3 cal_not_done . . . . .	257
6.136 vtss_restart_status_t Struct Reference . . . . .	258
6.136.1 Detailed Description . . . . .	258
6.136.2 Field Documentation . . . . .	258
6.136.2.1 restart . . . . .	258

6.136.2.2 prev_version . . . . .	258
6.136.2.3 cur_version . . . . .	259
6.137vtss_routing_entry_t Struct Reference . . . . .	259
6.137.1 Detailed Description . . . . .	259
6.137.2 Field Documentation . . . . .	259
6.137.2.1 type . . . . .	259
6.137.2.2 ipv4_uc . . . . .	260
6.137.2.3 ipv6_uc . . . . .	260
6.137.2.4 route . . . . .	260
6.137.2.5 vlan . . . . .	260
6.138vtss_secure_on_passwd_t Struct Reference . . . . .	260
6.138.1 Detailed Description . . . . .	261
6.138.2 Field Documentation . . . . .	261
6.138.2.1 passwd . . . . .	261
6.139vtss_serd़es_macro_conf_t Struct Reference . . . . .	261
6.139.1 Detailed Description . . . . .	261
6.139.2 Field Documentation . . . . .	261
6.139.2.1 serdes1g_vdd . . . . .	262
6.139.2.2 serdes6g_vdd . . . . .	262
6.139.2.3 ib_cterm_ena . . . . .	262
6.139.2.4 qsgmii . . . . .	262
6.140vtss_sflow_port_conf_t Struct Reference . . . . .	262
6.140.1 Detailed Description . . . . .	263
6.140.2 Field Documentation . . . . .	263
6.140.2.1 type . . . . .	263
6.140.2.2 sampling_rate . . . . .	263
6.141vtss_sgpio_conf_t Struct Reference . . . . .	263
6.141.1 Detailed Description . . . . .	264
6.141.2 Field Documentation . . . . .	264
6.141.2.1 bmode . . . . .	264

---

6.141.2.2 bit_count . . . . .	264
6.141.2.3 port_conf . . . . .	264
6.142vtss_sgpi_port_conf_t Struct Reference . . . . .	265
6.142.1 Detailed Description . . . . .	265
6.142.2 Field Documentation . . . . .	265
6.142.2.1 enabled . . . . .	265
6.142.2.2 mode . . . . .	265
6.142.2.3 int_pol_high . . . . .	266
6.143vtss_sgpi_port_data_t Struct Reference . . . . .	266
6.143.1 Detailed Description . . . . .	266
6.143.2 Field Documentation . . . . .	266
6.143.2.1 value . . . . .	266
6.144vtss_shaper_t Struct Reference . . . . .	267
6.144.1 Detailed Description . . . . .	267
6.144.2 Field Documentation . . . . .	267
6.144.2.1 level . . . . .	267
6.144.2.2 rate . . . . .	267
6.145vtss_sync_clock_in_t Struct Reference . . . . .	268
6.145.1 Detailed Description . . . . .	268
6.145.2 Field Documentation . . . . .	268
6.145.2.1 port_no . . . . .	268
6.145.2.2 squelsh . . . . .	268
6.145.2.3 enable . . . . .	269
6.146vtss_sync_clock_out_t Struct Reference . . . . .	269
6.146.1 Detailed Description . . . . .	269
6.146.2 Field Documentation . . . . .	269
6.146.2.1 divider . . . . .	269
6.146.2.2 enable . . . . .	270
6.147vtss_tci_t Struct Reference . . . . .	270
6.147.1 Detailed Description . . . . .	270

6.147.2 Field Documentation . . . . .	270
6.147.2.1 vid . . . . .	270
6.147.2.2 cfi . . . . .	271
6.147.2.3 tagprior . . . . .	271
6.148vtss_timeofday_t Struct Reference . . . . .	271
6.148.1 Detailed Description . . . . .	271
6.148.2 Field Documentation . . . . .	271
6.148.2.1 sec [1/2] . . . . .	272
6.148.2.2 sec [2/2] . . . . .	272
6.149vtss_timestamp_t Struct Reference . . . . .	272
6.149.1 Detailed Description . . . . .	272
6.149.2 Field Documentation . . . . .	272
6.149.2.1 sec_msb . . . . .	273
6.149.2.2 seconds . . . . .	273
6.149.2.3 nanoseconds . . . . .	273
6.150vtss_trace_conf_t Struct Reference . . . . .	273
6.150.1 Detailed Description . . . . .	273
6.150.2 Field Documentation . . . . .	274
6.150.2.1 level . . . . .	274
6.151vtss_ts_ext_clock_mode_t Struct Reference . . . . .	274
6.151.1 Detailed Description . . . . .	274
6.151.2 Field Documentation . . . . .	274
6.151.2.1 one_pps_mode . . . . .	274
6.151.2.2 enable . . . . .	275
6.151.2.3 freq . . . . .	275
6.152vtss_ts_id_t Struct Reference . . . . .	275
6.152.1 Detailed Description . . . . .	275
6.152.2 Field Documentation . . . . .	275
6.152.2.1 ts_id . . . . .	276
6.153vtss_ts_internal_mode_t Struct Reference . . . . .	276

6.153.1 Detailed Description	276
6.153.2 Field Documentation	276
6.153.2.1 int_fmt	276
6.154vtss_ts_operation_mode_t Struct Reference	277
6.154.1 Detailed Description	277
6.154.2 Field Documentation	277
6.154.2.1 mode	277
6.155vtss_ts_timestamp_alloc_t Struct Reference	277
6.155.1 Detailed Description	278
6.155.2 Field Documentation	278
6.155.2.1 port_mask	278
6.155.2.2 context	278
6.155.2.3 cb	278
6.156vtss_ts_timestamp_t Struct Reference	278
6.156.1 Detailed Description	279
6.156.2 Field Documentation	279
6.156.2.1 ts	279
6.156.2.2 id	279
6.156.2.3 context	279
6.156.2.4 ts_valid	280
6.157vtss_vcap_ip_t Struct Reference	280
6.157.1 Detailed Description	280
6.157.2 Field Documentation	280
6.157.2.1 value	280
6.157.2.2 mask	281
6.158vtss_vcap_u128_t Struct Reference	281
6.158.1 Detailed Description	281
6.158.2 Field Documentation	281
6.158.2.1 value	281
6.158.2.2 mask	282

6.159vtss_vcap_u16_t Struct Reference . . . . .	282
6.159.1 Detailed Description . . . . .	282
6.159.2 Field Documentation . . . . .	282
6.159.2.1 value . . . . .	282
6.159.2.2 mask . . . . .	283
6.160vtss_vcap_u24_t Struct Reference . . . . .	283
6.160.1 Detailed Description . . . . .	283
6.160.2 Field Documentation . . . . .	283
6.160.2.1 value . . . . .	283
6.160.2.2 mask . . . . .	284
6.161vtss_vcap_u32_t Struct Reference . . . . .	284
6.161.1 Detailed Description . . . . .	284
6.161.2 Field Documentation . . . . .	284
6.161.2.1 value . . . . .	284
6.161.2.2 mask . . . . .	285
6.162vtss_vcap_u40_t Struct Reference . . . . .	285
6.162.1 Detailed Description . . . . .	285
6.162.2 Field Documentation . . . . .	285
6.162.2.1 value . . . . .	285
6.162.2.2 mask . . . . .	286
6.163vtss_vcap_u48_t Struct Reference . . . . .	286
6.163.1 Detailed Description . . . . .	286
6.163.2 Field Documentation . . . . .	286
6.163.2.1 value . . . . .	286
6.163.2.2 mask . . . . .	287
6.164vtss_vcap_u8_t Struct Reference . . . . .	287
6.164.1 Detailed Description . . . . .	287
6.164.2 Field Documentation . . . . .	287
6.164.2.1 value . . . . .	287
6.164.2.2 mask . . . . .	288

---

6.165vtss_vcap_udp_tcp_t Struct Reference . . . . .	288
6.165.1 Detailed Description . . . . .	288
6.165.2 Field Documentation . . . . .	288
6.165.2.1 in_range . . . . .	288
6.165.2.2 low . . . . .	289
6.165.2.3 high . . . . .	289
6.166vtss_vcap_vid_t Struct Reference . . . . .	289
6.166.1 Detailed Description . . . . .	289
6.166.2 Field Documentation . . . . .	289
6.166.2.1 value . . . . .	290
6.166.2.2 mask . . . . .	290
6.167vtss_vcap_vr_t Struct Reference . . . . .	290
6.167.1 Detailed Description . . . . .	290
6.167.2 Field Documentation . . . . .	291
6.167.2.1 type . . . . .	291
6.167.2.2 value . . . . .	291
6.167.2.3 mask . . . . .	291
6.167.2.4 v . . . . .	291
6.167.2.5 low . . . . .	291
6.167.2.6 high . . . . .	292
6.167.2.7 r . . . . .	292
6.167.2.8 vr . . . . .	292
6.168vtss_vce_action_t Struct Reference . . . . .	292
6.168.1 Detailed Description . . . . .	292
6.168.2 Field Documentation . . . . .	292
6.168.2.1 vid . . . . .	293
6.168.2.2 policy_no . . . . .	293
6.169vtss_vce_frame_etype_t Struct Reference . . . . .	293
6.169.1 Detailed Description . . . . .	293
6.169.2 Field Documentation . . . . .	293

6.169.2.1 <code>etype</code>	294
6.169.2.2 <code>data</code>	294
6.170 <code>vtss_vce_frame_ipv4_t</code> Struct Reference	294
6.170.1 Detailed Description	294
6.170.2 Field Documentation	294
6.170.2.1 <code>fragment</code>	295
6.170.2.2 <code>options</code>	295
6.170.2.3 <code>dscp</code>	295
6.170.2.4 <code>proto</code>	295
6.170.2.5 <code>sip</code>	295
6.170.2.6 <code>dport</code>	296
6.171 <code>vtss_vce_frame_ipv6_t</code> Struct Reference	296
6.171.1 Detailed Description	296
6.171.2 Field Documentation	296
6.171.2.1 <code>dscp</code>	296
6.171.2.2 <code>proto</code>	297
6.171.2.3 <code>sip</code>	297
6.171.2.4 <code>dport</code>	297
6.172 <code>vtss_vce_frame_llc_t</code> Struct Reference	297
6.172.1 Detailed Description	297
6.172.2 Field Documentation	298
6.172.2.1 <code>data</code>	298
6.173 <code>vtss_vce_frame_snap_t</code> Struct Reference	298
6.173.1 Detailed Description	298
6.173.2 Field Documentation	298
6.173.2.1 <code>data</code>	298
6.174 <code>vtss_vce_key_t</code> Struct Reference	299
6.174.1 Detailed Description	299
6.174.2 Field Documentation	299
6.174.2.1 <code>port_list</code>	299

6.174.2.2 mac . . . . .	299
6.174.2.3 tag . . . . .	300
6.174.2.4 type . . . . .	300
6.174.2.5 etype . . . . .	300
6.174.2.6 llc . . . . .	300
6.174.2.7 snap . . . . .	300
6.174.2.8 ipv4 . . . . .	301
6.174.2.9 ipv6 . . . . .	301
6.174.2.10 frame . . . . .	301
6.175vtss_vce_mac_t Struct Reference . . . . .	301
6.175.1 Detailed Description . . . . .	301
6.175.2 Field Documentation . . . . .	302
6.175.2.1 dmac_mc . . . . .	302
6.175.2.2 dmac_bc . . . . .	302
6.175.2.3 smac . . . . .	302
6.176vtss_vce_t Struct Reference . . . . .	302
6.176.1 Detailed Description . . . . .	303
6.176.2 Field Documentation . . . . .	303
6.176.2.1 id . . . . .	303
6.176.2.2 key . . . . .	303
6.176.2.3 action . . . . .	303
6.177vtss_vce_tag_t Struct Reference . . . . .	303
6.177.1 Detailed Description . . . . .	304
6.177.2 Field Documentation . . . . .	304
6.177.2.1 vid . . . . .	304
6.177.2.2 pcp . . . . .	304
6.177.2.3 dei . . . . .	304
6.177.2.4 tagged . . . . .	305
6.177.2.5 s_tag . . . . .	305
6.178vtss_vcl_port_conf_t Struct Reference . . . . .	305

6.178.1 Detailed Description . . . . .	305
6.178.2 Field Documentation . . . . .	305
6.178.2.1 dmac_dip . . . . .	306
6.179vtss_vid_mac_t Struct Reference . . . . .	306
6.179.1 Detailed Description . . . . .	306
6.179.2 Field Documentation . . . . .	306
6.179.2.1 vid . . . . .	306
6.179.2.2 mac . . . . .	307
6.180vtss_vlan_conf_t Struct Reference . . . . .	307
6.180.1 Detailed Description . . . . .	307
6.180.2 Field Documentation . . . . .	307
6.180.2.1 s_etype . . . . .	307
6.181vtss_vlan_port_conf_t Struct Reference . . . . .	308
6.181.1 Detailed Description . . . . .	308
6.181.2 Field Documentation . . . . .	308
6.181.2.1 port_type . . . . .	308
6.181.2.2 pvid . . . . .	308
6.181.2.3 untagged_vid . . . . .	309
6.181.2.4 frame_type . . . . .	309
6.181.2.5 ingress_filter . . . . .	309
6.182vtss_vlan_tag_t Struct Reference . . . . .	309
6.182.1 Detailed Description . . . . .	309
6.182.2 Field Documentation . . . . .	310
6.182.2.1 tpid . . . . .	310
6.182.2.2 pcp . . . . .	310
6.182.2.3 dei . . . . .	310
6.182.2.4 vid . . . . .	310
6.183vtss_vlan_trans_grp2vlan_conf_t Struct Reference . . . . .	311
6.183.1 Detailed Description . . . . .	311
6.183.2 Field Documentation . . . . .	311

---

6.183.2.1 group_id . . . . .	311
6.183.2.2 vid . . . . .	311
6.183.2.3 trans_vid . . . . .	312
6.184vtss_vlan_trans_port2grp_conf_t Struct Reference . . . . .	312
6.184.1 Detailed Description . . . . .	312
6.184.2 Field Documentation . . . . .	312
6.184.2.1 group_id . . . . .	312
6.184.2.2 ports . . . . .	313
6.185vtss_vlan_vid_conf_t Struct Reference . . . . .	313
6.185.1 Detailed Description . . . . .	313
6.185.2 Field Documentation . . . . .	313
6.185.2.1 learning . . . . .	313
6.185.2.2 mirror . . . . .	314
6.186vtss_wol_mac_addr_t Struct Reference . . . . .	314
6.186.1 Detailed Description . . . . .	314
6.186.2 Field Documentation . . . . .	314
6.186.2.1 addr . . . . .	314
<b>7 File Documentation</b> . . . . .	<b>315</b>
7.1 vtss_api/include/vtss/api/l2_types.h File Reference . . . . .	315
7.1.1 Detailed Description . . . . .	315
7.1.2 Enumeration Type Documentation . . . . .	315
7.1.2.1 vtss_sflow_type_t . . . . .	315
7.2 vtss_api/include/vtss/api/options.h File Reference . . . . .	316
7.2.1 Detailed Description . . . . .	317
7.2.2 Macro Definition Documentation . . . . .	317
7.2.2.1 VTSS_ARCH_CARACAL . . . . .	318
7.2.2.2 VTSS_FEATURE_EVC . . . . .	318
7.2.2.3 VTSS_FEATURE_QOS_POLICER_DLB . . . . .	318
7.2.2.4 VTSS_ARCH_LUTON26 . . . . .	318
7.2.2.5 VTSS_FEATURE_MISC . . . . .	318

7.2.2.6	VTSS_FEATURE_SERIAL_GPIO . . . . .	319
7.2.2.7	VTSS_FEATURE_PORT_CONTROL . . . . .	319
7.2.2.8	VTSS_FEATURE_CLAUSE_37 . . . . .	319
7.2.2.9	VTSS_FEATURE_EXC_COL_CONT . . . . .	319
7.2.2.10	VTSS_FEATURE_PORT_CNT_BRIDGE . . . . .	319
7.2.2.11	VTSS_FEATURE_QOS . . . . .	320
7.2.2.12	VTSS_FEATURE_QCL . . . . .	320
7.2.2.13	VTSS_FEATURE_QCL_V2 . . . . .	320
7.2.2.14	VTSS_FEATURE_QCL_DMAC_DIP . . . . .	320
7.2.2.15	VTSS_FEATURE_QCL_KEY_S_TAG . . . . .	320
7.2.2.16	VTSS_FEATURE_QCL_PCP_DEI_ACTION . . . . .	321
7.2.2.17	VTSS_FEATURE_QCL_POLICY_ACTION . . . . .	321
7.2.2.18	VTSS_FEATURE_QOS_POLICER_UC_SWITCH . . . . .	321
7.2.2.19	VTSS_FEATURE_QOS_POLICER_MC_SWITCH . . . . .	321
7.2.2.20	VTSS_FEATURE_QOS_POLICER_BC_SWITCH . . . . .	321
7.2.2.21	VTSS_FEATURE_QOS_PORT_POLICER_EXT . . . . .	322
7.2.2.22	VTSS_FEATURE_QOS_PORT_POLICER_EXT_FPS . . . . .	322
7.2.2.23	VTSS_FEATURE_QOS_PORT_POLICER_EXT_FC . . . . .	322
7.2.2.24	VTSS_FEATURE_QOS_QUEUE_POLICER . . . . .	322
7.2.2.25	VTSS_FEATURE_QOS_QUEUE_TX . . . . .	322
7.2.2.26	VTSS_FEATURE_QOS_SCHEDULER_V2 . . . . .	323
7.2.2.27	VTSS_FEATURE_QOS_TAG_REMARK_V2 . . . . .	323
7.2.2.28	VTSS_FEATURE_QOS_CLASSIFICATION_V2 . . . . .	323
7.2.2.29	VTSS_FEATURE_QOS_EGRESS_QUEUE_SHAPERS . . . . .	323
7.2.2.30	VTSS_FEATURE_QOS_EGRESS_QUEUE_SHAPERS_EB . . . . .	323
7.2.2.31	VTSS_FEATURE_QOS_DSCP_CLASS_DP_AWARE . . . . .	324
7.2.2.32	VTSS_FEATURE_QOS_DSCP_REMARK . . . . .	324
7.2.2.33	VTSS_FEATURE_QOS_DSCP_REMARK_V2 . . . . .	324
7.2.2.34	VTSS_FEATURE_QOS_DSCP_REMARK_DP_AWARE . . . . .	324
7.2.2.35	VTSS_FEATURE_PACKET . . . . .	324

7.2.2.36 VTSS_FEATURE_PACKET_TX . . . . .	325
7.2.2.37 VTSS_FEATURE_PACKET_RX . . . . .	325
7.2.2.38 VTSS_FEATURE_PACKET_GROUPING . . . . .	325
7.2.2.39 VTSS_FEATURE_PACKET_PORT_REG . . . . .	325
7.2.2.40 VTSS_FEATURE_LAYER2 . . . . .	325
7.2.2.41 VTSS_FEATURE_PVLAN . . . . .	326
7.2.2.42 VTSS_FEATURE_VLAN_PORT_V2 . . . . .	326
7.2.2.43 VTSS_FEATURE_VLAN_TX_TAG . . . . .	326
7.2.2.44 VTSS_FEATURE_IPV4_MC_SIP . . . . .	326
7.2.2.45 VTSS_FEATURE_IPV6_MC_SIP . . . . .	326
7.2.2.46 VTSS_FEATURE_MAC_AGE_AUTO . . . . .	327
7.2.2.47 VTSS_FEATURE_MAC_CPU_QUEUE . . . . .	327
7.2.2.48 VTSS_FEATURE_EEE . . . . .	327
7.2.2.49 VTSS_FEATURE_PORT_MUX . . . . .	327
7.2.2.50 VTSS_FEATURE_FAN . . . . .	327
7.2.2.51 VTSS_FEATURE_VCAP [1/2] . . . . .	328
7.2.2.52 VTSS_FEATURE_ACL . . . . .	328
7.2.2.53 VTSS_FEATURE_ACL_V2 . . . . .	328
7.2.2.54 VTSS_FEATURE_VCL . . . . .	328
7.2.2.55 VTSS_FEATURE_TIMESTAMP . . . . .	328
7.2.2.56 VTSS_FEATURE_TIMESTAMP_ONE_STEP . . . . .	329
7.2.2.57 VTSS_FEATURE_TIMESTAMP_LATENCY_COMP . . . . .	329
7.2.2.58 VTSS_FEATURE_SYNCE . . . . .	329
7.2.2.59 VTSS_FEATURE_NPI . . . . .	329
7.2.2.60 VTSS_FEATURE_IRQ_CONTROL . . . . .	329
7.2.2.61 VTSS_FEATURE_LED_POW_REDUC . . . . .	330
7.2.2.62 VTSS_FEATURE_INTERRUPTS . . . . .	330
7.2.2.63 VTSS_FEATURE_VLAN_TRANSLATION . . . . .	330
7.2.2.64 VTSS_FEATURE_SFLOW . . . . .	330
7.2.2.65 VTSS_FEATURE_MIRROR_CPU . . . . .	330

7.2.2.66	VTSS_FEATURE_SERDES_MACRO_SETTINGS . . . . .	331
7.2.2.67	VTSS_CHIP CU PHY . . . . .	331
7.2.2.68	VTSS_OPT_TRACE . . . . .	331
7.2.2.69	VTSS_OPT_VAUI_EQ_CTRL . . . . .	331
7.2.2.70	VTSS_PHY_OPT_VERIPHY . . . . .	331
7.2.2.71	VTSS_FEATURE_WARM_START . . . . .	332
7.2.2.72	VTSS_FEATURE_VCAP [2/2] . . . . .	332
7.3	vtss_api/include/vtss/api/phy.h File Reference . . . . .	332
7.3.1	Detailed Description . . . . .	333
7.3.2	Macro Definition Documentation . . . . .	333
7.3.2.1	VTSS_PHY_POWER_ACTIPHYS_BIT . . . . .	333
7.3.2.2	VTSS_PHY_POWER_DYNAMIC_BIT . . . . .	333
7.3.3	Enumeration Type Documentation . . . . .	333
7.3.3.1	vtss_phy_power_mode_t . . . . .	333
7.3.3.2	vtss_phy_veriphy_status_t . . . . .	334
7.4	vtss_api/include/vtss/api/port.h File Reference . . . . .	334
7.4.1	Detailed Description . . . . .	336
7.4.2	Macro Definition Documentation . . . . .	336
7.4.2.1	PORT_CAP_NONE . . . . .	336
7.4.2.2	PORT_CAP_AUTONEG . . . . .	337
7.4.2.3	PORT_CAP_10M_HDX . . . . .	337
7.4.2.4	PORT_CAP_10M_FDX . . . . .	337
7.4.2.5	PORT_CAP_100M_HDX . . . . .	337
7.4.2.6	PORT_CAP_100M_FDX . . . . .	337
7.4.2.7	PORT_CAP_1G_FDX . . . . .	338
7.4.2.8	PORT_CAP_2_5G_FDX . . . . .	338
7.4.2.9	PORT_CAP_5G_FDX . . . . .	338
7.4.2.10	PORT_CAP_10G_FDX . . . . .	338
7.4.2.11	PORT_CAP_FLOW_CTRL . . . . .	338
7.4.2.12	PORT_CAP_COPPER . . . . .	339

7.4.2.13	PORT_CAP_FIBER	339
7.4.2.14	PORT_CAP_DUAL_COPPER	339
7.4.2.15	PORT_CAP_DUAL_FIBER	339
7.4.2.16	PORT_CAP_SD_ENABLE	339
7.4.2.17	PORT_CAP_SD_HIGH	340
7.4.2.18	PORT_CAP_SD_INTERNAL	340
7.4.2.19	PORT_CAP_DUAL_FIBER_100FX	340
7.4.2.20	PORT_CAP_XAUI_LANE_FLIP	340
7.4.2.21	PORT_CAP_VTSS_10G_PHY	340
7.4.2.22	PORT_CAP_SFP_DETECT	341
7.4.2.23	PORT_CAP_STACKING	341
7.4.2.24	PORT_CAP_DUAL_SFP_DETECT	341
7.4.2.25	PORT_CAP_SFP_ONLY	341
7.4.2.26	PORT_CAP_DUAL_COPPER_100FX	341
7.4.2.27	PORT_CAP_HDX	342
7.4.2.28	PORT_CAP_TRI_SPEED_FDX	342
7.4.2.29	PORT_CAP_TRI_SPEED	342
7.4.2.30	PORT_CAP_1G_PHY	342
7.4.2.31	PORT_CAP_TRI_SPEED_COPPER	342
7.4.2.32	PORT_CAP_TRI_SPEED_FIBER	343
7.4.2.33	PORT_CAP_TRI_SPEED_DUAL_COPPER	343
7.4.2.34	PORT_CAP_TRI_SPEED_DUAL_FIBER	343
7.4.2.35	PORT_CAP_TRI_SPEED_DUAL_FIBER_100FX	343
7.4.2.36	PORT_CAP_ANY_FIBER	343
7.4.2.37	PORT_CAP_SPEED_DUAL_ANY_FIBER_FIXED_SPEED	344
7.4.2.38	PORT_CAP_SPEED_DUAL_ANY_FIBER	344
7.4.2.39	PORT_CAP_TRI_SPEED_DUAL_ANY_FIBER	344
7.4.2.40	PORT_CAP_TRI_SPEED_DUAL_ANY_FIBER_FIXED_SFP_SPEED	344
7.4.2.41	PORT_CAP_DUAL_FIBER_1000X	344
7.4.2.42	PORT_CAP_SFP_1G	345

7.4.2.43	PORT_CAP_SFP_2_5G . . . . .	345
7.4.2.44	PORT_CAP_SFP_SD_HIGH . . . . .	345
7.4.2.45	PORT_CAP_2_5G_TRI_SPEED_FDX . . . . .	345
7.4.2.46	PORT_CAP_2_5G_TRI_SPEED . . . . .	345
7.4.2.47	PORT_CAP_2_5G_TRI_SPEED_COPPER . . . . .	346
7.4.3	Typedef Documentation . . . . .	346
7.4.3.1	port_cap_t . . . . .	346
7.4.4	Enumeration Type Documentation . . . . .	346
7.4.4.1	vtss_port_speed_t . . . . .	346
7.4.4.2	vtss_fiber_port_speed_t . . . . .	347
7.5	vtss_api/include/vtss/api/types.h File Reference . . . . .	347
7.5.1	Detailed Description . . . . .	354
7.5.2	Macro Definition Documentation . . . . .	354
7.5.2.1	PRIu64 . . . . .	354
7.5.2.2	PRIi64 . . . . .	354
7.5.2.3	PRIx64 . . . . .	355
7.5.2.4	VTSS_BIT64 . . . . .	355
7.5.2.5	VTSS_BITMASK64 . . . . .	355
7.5.2.6	VTSS_EXTRACT_BITFIELD64 . . . . .	355
7.5.2.7	VTSS_ENCODE_BITFIELD64 . . . . .	355
7.5.2.8	VTSS_ENCODE_BITMASK64 . . . . .	356
7.5.2.9	TRUE . . . . .	356
7.5.2.10	FALSE . . . . .	356
7.5.2.11	VTSS_PACKET_RATE_DISABLED . . . . .	356
7.5.2.12	VTSS_PORT_COUNT [1/2] . . . . .	356
7.5.2.13	VTSS_PORT_COUNT [2/2] . . . . .	357
7.5.2.14	VTSS_PORTS . . . . .	357
7.5.2.15	VTSS_PORT_NO_NONE . . . . .	357
7.5.2.16	VTSS_PORT_NO_CPU . . . . .	357
7.5.2.17	VTSS_PORT_NO_START . . . . .	357

7.5.2.18	VTSS_PORT_NO_END . . . . .	358
7.5.2.19	VTSS_PORT_ARRAY_SIZE . . . . .	358
7.5.2.20	VTSS_PORT_IS_PORT . . . . .	358
7.5.2.21	VTSS_PRIOS . . . . .	358
7.5.2.22	VTSS_PRIO_NO_NONE . . . . .	358
7.5.2.23	VTSS_PRIO_START . . . . .	359
7.5.2.24	VTSS_PRIO_END . . . . .	359
7.5.2.25	VTSS_PRIO_ARRAY_SIZE . . . . .	359
7.5.2.26	VTSS_QUEUES . . . . .	359
7.5.2.27	VTSS_QUEUE_START . . . . .	359
7.5.2.28	VTSS_QUEUE_END . . . . .	360
7.5.2.29	VTSS_QUEUE_ARRAY_SIZE . . . . .	360
7.5.2.30	VTSS_PCPS . . . . .	360
7.5.2.31	VTSS_PCP_START . . . . .	360
7.5.2.32	VTSS_PCP_END . . . . .	360
7.5.2.33	VTSS_PCP_ARRAY_SIZE . . . . .	361
7.5.2.34	VTSS_DEIS . . . . .	361
7.5.2.35	VTSS_DEI_START . . . . .	361
7.5.2.36	VTSS_DEI_END . . . . .	361
7.5.2.37	VTSS_DEI_ARRAY_SIZE . . . . .	361
7.5.2.38	VTSS_DPLS . . . . .	362
7.5.2.39	VTSS_DPL_START . . . . .	362
7.5.2.40	VTSS_DPL_END . . . . .	362
7.5.2.41	VTSS_DPL_ARRAY_SIZE . . . . .	362
7.5.2.42	VTSS_BITRATE_DISABLED . . . . .	362
7.5.2.43	VTSS_VID_NULL . . . . .	363
7.5.2.44	VTSS_VID_DEFAULT . . . . .	363
7.5.2.45	VTSS_VID_RESERVED . . . . .	363
7.5.2.46	VTSS_VIDS . . . . .	363
7.5.2.47	VTSS_VID_ALL . . . . .	363

7.5.2.48	VTSS_ETYPE_VTSS . . . . .	364
7.5.2.49	VTSS_MAC_ADDR_SZ_BYTES . . . . .	364
7.5.2.50	MAC_ADDR_BROADCAST . . . . .	364
7.5.2.51	VTSS_EVCS . . . . .	364
7.5.2.52	VTSS_ISDX_NONE . . . . .	364
7.5.2.53	VTSS_AGGRS . . . . .	365
7.5.2.54	VTSS_AGGR_NO_NONE . . . . .	365
7.5.2.55	VTSS_AGGR_NO_START . . . . .	365
7.5.2.56	VTSS_AGGR_NO_END . . . . .	365
7.5.2.57	VTSS_GLAGS . . . . .	365
7.5.2.58	VTSS_GLAG_NO_NONE . . . . .	366
7.5.2.59	VTSS_GLAG_NO_START . . . . .	366
7.5.2.60	VTSS_GLAG_NO_END . . . . .	366
7.5.2.61	VTSS_GLAG_PORTS . . . . .	366
7.5.2.62	VTSS_GLAG_PORT_START . . . . .	366
7.5.2.63	VTSS_GLAG_PORT_END . . . . .	367
7.5.2.64	VTSS_GLAG_PORT_ARRAY_SIZE . . . . .	367
7.5.2.65	VTSS_PACKET_RX_QUEUE_CNT . . . . .	367
7.5.2.66	VTSS_PACKET_RX_GRP_CNT . . . . .	367
7.5.2.67	VTSS_PACKET_TX_GRP_CNT . . . . .	367
7.5.2.68	VTSS_PACKET_RX_QUEUE_NONE . . . . .	368
7.5.2.69	VTSS_PACKET_RX_QUEUE_START . . . . .	368
7.5.2.70	VTSS_PACKET_RX_QUEUE_END . . . . .	368
7.5.2.71	VTSS_ACL_POLICERS . . . . .	368
7.5.2.72	VTSS_ACL_POLICER_NO_START . . . . .	368
7.5.2.73	VTSS_ACL_POLICER_NO_END . . . . .	369
7.5.2.74	VTSS_ACL_POLICY_NO_NONE . . . . .	369
7.5.2.75	VTSS_ACL_POLICY_NO_MIN . . . . .	369
7.5.2.76	VTSS_ACL_POLICY_NO_MAX . . . . .	369
7.5.2.77	VTSS_ACL_POLICIES . . . . .	369

7.5.2.78	VTSS_ACL_POLICY_NO_START . . . . .	370
7.5.2.79	VTSS_ACL_POLICY_NO_END . . . . .	370
7.5.2.80	VTSS_HQOS_COUNT . . . . .	370
7.5.2.81	VTSS_HQOS_ID_NONE . . . . .	370
7.5.2.82	VTSS_ONE_MIA . . . . .	370
7.5.2.83	VTSS_ONE_MILL . . . . .	371
7.5.2.84	VTSS_MAX_TIMEINTERVAL . . . . .	371
7.5.2.85	VTSS_INTERVAL_SEC . . . . .	371
7.5.2.86	VTSS_INTERVAL_MS . . . . .	371
7.5.2.87	VTSS_INTERVAL_US . . . . .	371
7.5.2.88	VTSS_INTERVAL_NS . . . . .	372
7.5.2.89	VTSS_INTERVAL_PS . . . . .	372
7.5.2.90	VTSS_SEC_NS_INTERVAL . . . . .	372
7.5.2.91	VTSS_CLOCK_IDENTITY_LENGTH . . . . .	372
7.5.2.92	VTSS_SYNCE_CLK_PORT_ARRAY_SIZE . . . . .	372
7.5.3	Typedef Documentation . . . . .	373
7.5.3.1	i8 . . . . .	373
7.5.3.2	i16 . . . . .	373
7.5.3.3	i32 . . . . .	373
7.5.3.4	i64 . . . . .	373
7.5.3.5	u8 . . . . .	374
7.5.3.6	u16 . . . . .	374
7.5.3.7	u32 . . . . .	374
7.5.3.8	u64 . . . . .	374
7.5.3.9	BOOL . . . . .	374
7.5.3.10	uintptr_t . . . . .	375
7.5.3.11	vtss_mac_addr_t . . . . .	375
7.5.3.12	vtss_isdx_t . . . . .	375
7.5.3.13	vtss_packet_rx_grp_t . . . . .	375
7.5.3.14	vtss_packet_tx_grp_t . . . . .	375

7.5.4	Enumeration Type Documentation	375
7.5.4.1	anonymous enum	375
7.5.4.2	vtss_mem_flags_t	378
7.5.4.3	vtss_port_interface_t	378
7.5.4.4	vtss_serdes_mode_t	379
7.5.4.5	vtss_storm_policer_mode_t	380
7.5.4.6	vtss_policer_type_t	380
7.5.4.7	vtss_vlan_frame_t	380
7.5.4.8	vtss_packet_reg_type_t	381
7.5.4.9	vtss_vdd_t	381
7.5.4.10	vtss_ip_type_t	381
7.5.4.11	vtss_vcap_bit_t	382
7.5.4.12	vtss_vcap_vr_type_t	382
7.5.4.13	vtss_vcap_key_type_t	382
7.5.4.14	vtss_ece_dir_t	383
7.5.4.15	vtss_ece_pop_tag_t	383
7.5.4.16	vtss_hqos_sch_mode_t	383
7.6	vtss_api/include/vtss_ae_api.h File Reference	384
7.6.1	Detailed Description	384
7.7	vtss_api/include/vtss_afi_api.h File Reference	384
7.7.1	Detailed Description	384
7.8	vtss_api/include/vtss_aneg_api.h File Reference	384
7.8.1	Detailed Description	384
7.9	vtss_api/include/vtss_api.h File Reference	385
7.9.1	Detailed Description	385
7.10	vtss_api/include/vtss_evc_api.h File Reference	385
7.10.1	Detailed Description	387
7.10.2	Macro Definition Documentation	387
7.10.2.1	VTSS_EVC_POLICERS	387
7.10.2.2	VTSS_EVC_ID_NONE	388

7.10.2.3	VTSS_ECE_ID_LAST	388
7.10.2.4	VTSS_MCE_ID_LAST	388
7.10.2.5	VTSS_MCE_POP_NONE	388
7.10.3	Enumeration Type Documentation	388
7.10.3.1	vtss_evc_vid_mode_t	388
7.10.3.2	vtss_evc_inner_tag_type_t	389
7.10.3.3	vtss_ece_type_t	389
7.10.3.4	vtss_ece_port_t	389
7.10.4	Function Documentation	390
7.10.4.1	vtss_evc_port_conf_get()	390
7.10.4.2	vtss_evc_port_conf_set()	390
7.10.4.3	vtss_evc_policer_conf_get()	391
7.10.4.4	vtss_evc_policer_conf_set()	391
7.10.4.5	vtss_evc_add()	392
7.10.4.6	vtss_evc_del()	392
7.10.4.7	vtss_evc_get()	392
7.10.4.8	vtss_ece_init()	393
7.10.4.9	vtss_ece_add()	393
7.10.4.10	vtss_ece_del()	394
7.10.4.11	vtss_mce_init()	394
7.10.4.12	vtss_mce_add()	394
7.10.4.13	vtss_mce_del()	395
7.11	vtss_api/include/vtss_fdma_api.h File Reference	395
7.11.1	Detailed Description	395
7.12	vtss_api/include/vtss_gfp_api.h File Reference	396
7.12.1	Detailed Description	396
7.13	vtss_api/include/vtss_hqos_api.h File Reference	396
7.13.1	Detailed Description	396
7.14	vtss_api/include/vtss_i2c_api.h File Reference	396
7.14.1	Detailed Description	396

7.15 vtss_api/include/vtss_init_api.h File Reference . . . . .	397
7.15.1 Detailed Description . . . . .	399
7.15.2 Macro Definition Documentation . . . . .	399
7.15.2.1 VTSS_I2C_NO_MULTIPLEXER . . . . .	399
7.15.3 Typedef Documentation . . . . .	399
7.15.3.1 vtss_reg_read_t . . . . .	399
7.15.3.2 vtss_reg_write_t . . . . .	400
7.15.3.3 vtss_i2c_read_t . . . . .	400
7.15.3.4 vtss_i2c_write_t . . . . .	401
7.15.3.5 vtss_spi_read_write_t . . . . .	401
7.15.3.6 vtss_spi_32bit_read_write_t . . . . .	402
7.15.3.7 vtss_spi_64bit_read_write_t . . . . .	402
7.15.3.8 vtss_miim_read_t . . . . .	403
7.15.3.9 vtss_miim_write_t . . . . .	403
7.15.3.10 vtss_mmd_read_t . . . . .	404
7.15.3.11 vtss_mmd_read_inc_t . . . . .	404
7.15.3.12 vtss_mmd_write_t . . . . .	405
7.15.4 Enumeration Type Documentation . . . . .	405
7.15.4.1 vtss_target_type_t . . . . .	405
7.15.4.2 vtss_port_mux_mode_t . . . . .	406
7.15.4.3 vtss_restart_t . . . . .	406
7.15.5 Function Documentation . . . . .	407
7.15.5.1 vtss_inst_get() . . . . .	407
7.15.5.2 vtss_inst_create() . . . . .	407
7.15.5.3 vtss_inst_destroy() . . . . .	408
7.15.5.4 vtss_init_conf_get() . . . . .	408
7.15.5.5 vtss_init_conf_set() . . . . .	408
7.15.5.6 vtss_restart_conf_end() . . . . .	409
7.15.5.7 vtss_restart_status_get() . . . . .	409
7.15.5.8 vtss_restart_conf_get() . . . . .	410

7.15.5.9 vtss_restart_conf_set()	410
7.16 vtss_api/include/vtss_l2_api.h File Reference	410
7.16.1 Detailed Description	418
7.16.2 Macro Definition Documentation	418
7.16.2.1 VTSS_MAC_ADDRS	418
7.16.2.2 VTSS_MSTIS	418
7.16.2.3 VTSS_MSTI_START	418
7.16.2.4 VTSS_MSTI_END	418
7.16.2.5 VTSS_MSTI_ARRAY_SIZE	419
7.16.2.6 VTSS_VCL_IDS	419
7.16.2.7 VTSS_VCL_ID_START	419
7.16.2.8 VTSS_VCL_ID_END	419
7.16.2.9 VTSS_VCL_ARRAY_SIZE	419
7.16.2.10 VTSS_VCE_ID_LAST	420
7.16.2.11 VTSS_VLAN_TRANS_GROUP_MAX_CNT	420
7.16.2.12 VTSS_VLAN_TRANS_MAX_CNT	420
7.16.2.13 VTSS_VLAN_TRANS_NULL_GROUP_ID	420
7.16.2.14 VTSS_VLAN_TRANS_FIRST_GROUP_ID	420
7.16.2.15 VTSS_VLAN_TRANS_VID_START	421
7.16.2.16 VTSS_VLAN_TRANS_MAX_VLAN_ID	421
7.16.2.17 VTSS_VLAN_TRANS_LAST_GROUP_ID	421
7.16.2.18 VTSS_VLAN_TRANS_VALID_GROUP_CHECK	421
7.16.2.19 VTSS_VLAN_TRANS_VALID_VLAN_CHECK	422
7.16.2.20 VTSS_VLAN_TRANS_NULL_CHECK	422
7.16.2.21 VTSS_VLAN_TRANS_PORT_BF_SIZE	422
7.16.2.22 VTSS_PVLANS	422
7.16.2.23 VTSS_PVLAN_NO_START	423
7.16.2.24 VTSS_PVLAN_NO_END	423
7.16.2.25 VTSS_PVLAN_ARRAY_SIZE	423
7.16.2.26 VTSS_PVLAN_NO_DEFAULT	423

7.16.2.27 VTSS_ERPI_S...	423
7.16.2.28 VTSS_ERPI_START . . . . .	424
7.16.2.29 VTSS_ERPI_END . . . . .	424
7.16.2.30 VTSS_ERPI_ARRAY_SIZE . . . . .	424
7.16.3 Typedef Documentation . . . . .	424
7.16.3.1 vtss_vt_id_t . . . . .	424
7.16.4 Enumeration Type Documentation . . . . .	424
7.16.4.1 vtss_stp_state_t . . . . .	424
7.16.4.2 vtss_vlan_port_type_t . . . . .	425
7.16.4.3 vtss_vlan_tx_tag_t . . . . .	425
7.16.4.4 vtss_vce_type_t . . . . .	425
7.16.4.5 vtss_eps_port_type_t . . . . .	426
7.16.4.6 vtss_eps_selector_t . . . . .	426
7.16.4.7 vtss_erps_state_t . . . . .	426
7.16.5 Function Documentation . . . . .	427
7.16.5.1 vtss_mac_table_add() . . . . .	427
7.16.5.2 vtss_mac_table_del() . . . . .	427
7.16.5.3 vtss_mac_table_get() . . . . .	428
7.16.5.4 vtss_mac_table_get_next() . . . . .	428
7.16.5.5 vtss_mac_table_age_time_get() . . . . .	428
7.16.5.6 vtss_mac_table_age_time_set() . . . . .	429
7.16.5.7 vtss_mac_table_age() . . . . .	429
7.16.5.8 vtss_mac_table_vlan_age() . . . . .	430
7.16.5.9 vtss_mac_table_flush() . . . . .	430
7.16.5.10 vtss_mac_table_port_flush() . . . . .	430
7.16.5.11 vtss_mac_table_vlan_flush() . . . . .	431
7.16.5.12 vtss_mac_table_vlan_port_flush() . . . . .	431
7.16.5.13 vtss_mac_table_status_get() . . . . .	431
7.16.5.14 vtss_learn_port_mode_get() . . . . .	432
7.16.5.15 vtss_learn_port_mode_set() . . . . .	432

7.16.5.16 vtss_port_state_get()	433
7.16.5.17 vtss_port_state_set()	433
7.16.5.18 vtss_stp_port_state_get()	434
7.16.5.19 vtss_stp_port_state_set()	434
7.16.5.20 vtss_mstp_vlan_msti_get()	434
7.16.5.21 vtss_mstp_vlan_msti_set()	435
7.16.5.22 vtss_mstp_port_msti_state_get()	435
7.16.5.23 vtss_mstp_port_msti_state_set()	436
7.16.5.24 vtss_vlan_conf_get()	436
7.16.5.25 vtss_vlan_conf_set()	437
7.16.5.26 vtss_vlan_port_conf_get()	437
7.16.5.27 vtss_vlan_port_conf_set()	437
7.16.5.28 vtss_vlan_port_members_get()	438
7.16.5.29 vtss_vlan_port_members_set()	438
7.16.5.30 vtss_vlan_vid_conf_get()	439
7.16.5.31 vtss_vlan_vid_conf_set()	439
7.16.5.32 vtss_vlan_tx_tag_get()	440
7.16.5.33 vtss_vlan_tx_tag_set()	440
7.16.5.34 vtss_vcl_port_conf_get()	440
7.16.5.35 vtss_vcl_port_conf_set()	441
7.16.5.36 vtss_vce_init()	441
7.16.5.37 vtss_vce_add()	442
7.16.5.38 vtss_vce_del()	442
7.16.5.39 vtss_vlan_trans_group_add()	443
7.16.5.40 vtss_vlan_trans_group_del()	443
7.16.5.41 vtss_vlan_trans_group_get()	443
7.16.5.42 vtss_vlan_trans_group_to_port_set()	444
7.16.5.43 vtss_vlan_trans_group_to_port_get()	444
7.16.5.44 vtss_isolated_vlan_get()	445
7.16.5.45 vtss_isolated_vlan_set()	445

7.16.5.46 vtss_isolated_port_members_get()	446
7.16.5.47 vtss_isolated_port_members_set()	446
7.16.5.48 vtss_pvlan_port_members_get()	446
7.16.5.49 vtss_pvlan_port_members_set()	447
7.16.5.50 vtss_apvlan_port_members_get()	447
7.16.5.51 vtss_apvlan_port_members_set()	448
7.16.5.52 vtss_dgroup_port_conf_get()	448
7.16.5.53 vtss_dgroup_port_conf_set()	449
7.16.5.54 vtss_sflow_port_conf_get()	449
7.16.5.55 vtss_sflow_port_conf_set()	449
7.16.5.56 vtss_sflow_sampling_rate_convert()	450
7.16.5.57 vtss_aggr_port_members_get()	450
7.16.5.58 vtss_aggr_port_members_set()	451
7.16.5.59 vtss_aggr_mode_get()	451
7.16.5.60 vtss_aggr_mode_set()	452
7.16.5.61 vtss_mirror_conf_get()	452
7.16.5.62 vtss_mirror_conf_set()	452
7.16.5.63 vtss_mirror_monitor_port_get()	453
7.16.5.64 vtss_mirror_monitor_port_set()	453
7.16.5.65 vtss_mirror_ingress_ports_get()	454
7.16.5.66 vtss_mirror_ingress_ports_set()	454
7.16.5.67 vtss_mirror_egress_ports_get()	454
7.16.5.68 vtss_mirror_egress_ports_set()	455
7.16.5.69 vtss_mirror_cpu_ingress_get()	455
7.16.5.70 vtss_mirror_cpu_ingress_set()	455
7.16.5.71 vtss_mirror_cpu_egress_get()	456
7.16.5.72 vtss_mirror_cpu_egress_set()	456
7.16.5.73 vtss_uc_flood_members_get()	457
7.16.5.74 vtss_uc_flood_members_set()	457
7.16.5.75 vtss_mc_flood_members_get()	457

7.16.5.76 <code>vtss_mc_flood_members_set()</code>	458
7.16.5.77 <code>vtss_ipv4_mc_flood_members_get()</code>	458
7.16.5.78 <code>vtss_ipv4_mc_flood_members_set()</code>	459
7.16.5.79 <code>vtss_ipv4_mc_add()</code>	459
7.16.5.80 <code>vtss_ipv4_mc_del()</code>	459
7.16.5.81 <code>vtss_ipv6_mc_flood_members_get()</code>	461
7.16.5.82 <code>vtss_ipv6_mc_flood_members_set()</code>	461
7.16.5.83 <code>vtss_ipv6_mc_ctrl_flood_get()</code>	462
7.16.5.84 <code>vtss_ipv6_mc_ctrl_flood_set()</code>	462
7.16.5.85 <code>vtss_ipv6_mc_add()</code>	462
7.16.5.86 <code>vtss_ipv6_mc_del()</code>	463
7.16.5.87 <code>vtss_eps_port_conf_get()</code>	463
7.16.5.88 <code>vtss_eps_port_conf_set()</code>	464
7.16.5.89 <code>vtss_eps_port_selector_get()</code>	464
7.16.5.90 <code>vtss_eps_port_selector_set()</code>	465
7.16.5.91 <code>vtss_erps_vlan_member_get()</code>	465
7.16.5.92 <code>vtss_erps_vlan_member_set()</code>	465
7.16.5.93 <code>vtss_erps_port_state_get()</code>	467
7.16.5.94 <code>vtss_erps_port_state_set()</code>	467
7.17 <code>vtss_api/include/vtss_l3_api.h</code> File Reference	468
7.17.1 Detailed Description	468
7.18 <code>vtss_api/include/vtss_mac10g_api.h</code> File Reference	468
7.18.1 Detailed Description	468
7.19 <code>vtss_api/include/vtss_misc_api.h</code> File Reference	468
7.19.1 Detailed Description	474
7.19.2 Macro Definition Documentation	474
7.19.2.1 <code>VTSS_OS_TIMESTAMP_TYPE</code>	474
7.19.2.2 <code>VTSS_OS_TIMESTAMP</code>	474
7.19.3 Typedef Documentation	475
7.19.3.1 <code>tod_get_ns_cnt_cb_t</code>	475

7.19.4.4 Enumeration Type Documentation . . . . .	475
7.19.4.1 <code>vtss_trace_layer_t</code> . . . . .	475
7.19.4.2 <code>vtss_trace_group_t</code> . . . . .	475
7.19.4.3 <code>vtss_trace_level_t</code> . . . . .	476
7.19.4.4 <code>vtss_debug_layer_t</code> . . . . .	476
7.19.4.5 <code>vtss_debug_group_t</code> . . . . .	478
7.19.4.6 <code>vtss_gpio_mode_t</code> . . . . .	479
7.19.4.7 <code>vtss_sgpio_mode_t</code> . . . . .	479
7.19.4.8 <code>vtss_sgpio_bmode_t</code> . . . . .	480
7.19.4.9 <code>vtss_irq_t</code> . . . . .	480
7.19.4.10 <code>vtss_eee_state_select_t</code> . . . . .	480
7.19.5 Function Documentation . . . . .	481
7.19.5.1 <code>vtss_trace_conf_get()</code> . . . . .	481
7.19.5.2 <code>vtss_trace_conf_set()</code> . . . . .	481
7.19.5.3 <code>vtss_callout_trace_printf()</code> . . . . .	482
7.19.5.4 <code>vtss_callout_trace_hex_dump()</code> . . . . .	482
7.19.5.5 <code>vtss_debug_info_get()</code> . . . . .	483
7.19.5.6 <code>vtss_debug_info_print()</code> . . . . .	483
7.19.5.7 <code>vtss_callout_lock()</code> . . . . .	484
7.19.5.8 <code>vtss_callout_unlock()</code> . . . . .	484
7.19.5.9 <code>vtss_debug_lock()</code> . . . . .	484
7.19.5.10 <code>vtss_debug_unlock()</code> . . . . .	485
7.19.5.11 <code>vtss_reg_read()</code> . . . . .	485
7.19.5.12 <code>vtss_reg_write()</code> . . . . .	485
7.19.5.13 <code>vtss_reg_write_masked()</code> . . . . .	486
7.19.5.14 <code>vtss_intr_sticky_clear()</code> . . . . .	486
7.19.5.15 <code>vtss_chip_id_get()</code> . . . . .	487
7.19.5.16 <code>vtss_poll_1sec()</code> . . . . .	487
7.19.5.17 <code>vtss_ptp_event_poll()</code> . . . . .	488
7.19.5.18 <code>vtss_ptp_event_enable()</code> . . . . .	488

7.19.5.19 vtss_dev_all_event_poll()	488
7.19.5.20 vtss_dev_all_event_enable()	489
7.19.5.21 vtss_gpio_mode_set()	489
7.19.5.22 vtss_gpio_direction_set()	490
7.19.5.23 vtss_gpio_read()	490
7.19.5.24 vtss_gpio_write()	491
7.19.5.25 vtss_gpio_event_poll()	491
7.19.5.26 vtss_gpio_event_enable()	492
7.19.5.27 vtss_sgpio_conf_get()	492
7.19.5.28 vtss_sgpio_conf_set()	493
7.19.5.29 vtss_sgpio_read()	493
7.19.5.30 vtss_sgpio_event_poll()	493
7.19.5.31 vtss_sgpio_event_enable()	494
7.19.5.32 vtss_intr_cfg()	495
7.19.5.33 vtss_intr_mask_set()	495
7.19.5.34 vtss_intr_status_get()	495
7.19.5.35 vtss_intr_pol_negation()	496
7.19.5.36 vtss_irq_conf_get()	496
7.19.5.37 vtss_irq_conf_set()	497
7.19.5.38 vtss_irq_status_get_and_mask()	497
7.19.5.39 vtss_irq_enable()	497
7.19.5.40 vtss_tod_get_ns_cnt()	498
7.19.5.41 vtss_tod_set_ns_cnt_cb()	498
7.19.5.42 vtss_temp_sensor_init()	498
7.19.5.43 vtss_temp_sensor_get()	499
7.19.5.44 vtss_fan_rotation_get()	499
7.19.5.45 vtss_fan_cool_lvl_set()	500
7.19.5.46 vtss_fan_controller_init()	500
7.19.5.47 vtss_fan_cool_lvl_get()	500
7.19.5.48 vtss_eee_port_conf_set()	501

7.19.5.49 <code>vtss_eee_port_state_set()</code>	501
7.19.5.50 <code>vtss_eee_port_counter_get()</code>	502
7.19.5.51 <code>vtss_debug_reg_check_set()</code>	502
7.20 <code>vtss_api/include/vtss_mpls_api.h</code> File Reference	503
7.20.1 Detailed Description	503
7.21 <code>vtss_api/include/vtss_oam_api.h</code> File Reference	503
7.21.1 Detailed Description	503
7.22 <code>vtss_api/include/vtss_oha_api.h</code> File Reference	503
7.22.1 Detailed Description	504
7.23 <code>vtss_api/include/vtss_os.h</code> File Reference	504
7.23.1 Detailed Description	504
7.24 <code>vtss_api/include/vtss_os_custom.h</code> File Reference	504
7.24.1 Detailed Description	505
7.24.2 Macro Definition Documentation	505
7.24.2.1 <code>uint</code>	505
7.24.2.2 <code>ulong</code>	505
7.24.2.3 <code>VTSS_MSLEEP</code>	505
7.24.2.4 <code>VTSS_MTIMER_START</code>	505
7.24.2.5 <code>VTSS_MTIMER_TIMEOUT</code>	506
7.24.2.6 <code>VTSS_MTIMER_CANCEL</code>	506
7.24.2.7 <code>VTSS_DIV64</code>	506
7.24.2.8 <code>VTSS_MOD64</code>	506
7.24.2.9 <code>VTSS_LABS</code>	507
7.24.2.10 <code>VTSS_LLABS</code>	507
7.24.2.11 <code>VTSS_OS_CTZ</code>	507
7.24.2.12 <code>VTSS_OS_CTZ64</code>	507
7.24.2.13 <code>VTSS_OS_MALLOC</code>	508
7.24.2.14 <code>VTSS_OS_FREE</code>	508
7.24.2.15 <code>VTSS_OS RAND</code>	508
7.24.3 TypeDef Documentation	508

---

7.24.3.1 vtss_mtimer_t . . . . .	508
7.25 vtss_api/include/vtss_os_ecos.h File Reference . . . . .	509
7.25.1 Detailed Description . . . . .	510
7.25.2 Macro Definition Documentation . . . . .	510
7.25.2.1 VTSS_MSLEEP . . . . .	510
7.25.2.2 VTSS_NSLEEP . . . . .	510
7.25.2.3 VTSS_MTIMER_START . . . . .	511
7.25.2.4 VTSS_MTIMER_TIMEOUT . . . . .	511
7.25.2.5 VTSS_MTIMER_CANCEL . . . . .	511
7.25.2.6 VTSS_TIME_OF_DAY . . . . .	511
7.25.2.7 VTSS_DIV64 . . . . .	511
7.25.2.8 VTSS_MOD64 . . . . .	512
7.25.2.9 VTSS_LABS . . . . .	512
7.25.2.10 VTSS_LLABS . . . . .	512
7.25.2.11 VTSS_OS_CTZ . . . . .	512
7.25.2.12 VTSS_OS_CTZ64 . . . . .	513
7.25.2.13 VTSS_OS_MALLOC . . . . .	513
7.25.2.14 VTSS_OS_FREE . . . . .	513
7.25.2.15 VTSS_OS_RAND . . . . .	514
7.25.2.16 VTSS_OS_REORDER_BARRIER . . . . .	514
7.25.2.17 VTSS_OS_COMPILER_ATTRIBUTE_ALIGNED . . . . .	514
7.25.2.18 VTSS_OS_DCACHE_LINE_SIZE_BYTES . . . . .	514
7.25.2.19 VTSS_OS_DCACHE_INVALIDATE . . . . .	515
7.25.2.20 VTSS_OS_DCACHE_FLUSH . . . . .	515
7.25.2.21 VTSS_OS_VIRT_TO_PHYS . . . . .	515
7.25.2.22 VTSS_OS_BIG_ENDIAN . . . . .	515
7.25.2.23 VTSS_OS_NTOHL . . . . .	516
7.25.2.24 VTSS_OS_SCHEDULER_FLAGS . . . . .	516
7.25.2.25 VTSS_OS_SCHEDULER_LOCK . . . . .	516
7.25.2.26 VTSS_OS_SCHEDULER_UNLOCK . . . . .	516

7.25.2.27 VTSS_OS_INTERRUPT_FLAGS . . . . .	517
7.25.2.28 VTSS_OS_INTERRUPT_DISABLE . . . . .	517
7.25.2.29 VTSS_OS_INTERRUPT_RESTORE . . . . .	517
7.25.3 Typedef Documentation . . . . .	517
7.25.3.1 vtss_mtimer_t . . . . .	517
7.25.4 Function Documentation . . . . .	517
7.25.4.1 llabs() . . . . .	517
7.25.4.2 vtss_callout_malloc() . . . . .	518
7.25.4.3 vtss_callout_free() . . . . .	518
7.26 vtss_api/include/vtss_os_linux.h File Reference . . . . .	519
7.26.1 Detailed Description . . . . .	519
7.26.2 Macro Definition Documentation . . . . .	520
7.26.2.1 VTSS_OS_BIG_ENDIAN . . . . .	520
7.26.2.2 VTSS_OS_NTOHL . . . . .	520
7.26.2.3 VTSS_NSLEEP . . . . .	520
7.26.2.4 VTSS_MSLEEP . . . . .	521
7.26.2.5 VTSS_MTIMER_START . . . . .	521
7.26.2.6 VTSS_MTIMER_TIMEOUT . . . . .	521
7.26.2.7 VTSS_MTIMER_CANCEL . . . . .	522
7.26.2.8 VTSS_TIME_OF_DAY . . . . .	522
7.26.2.9 VTSS_OS_SCHEDULER_FLAGS . . . . .	522
7.26.2.10 VTSS_OS_SCHEDULER_LOCK . . . . .	522
7.26.2.11 VTSS_OS_SCHEDULER_UNLOCK . . . . .	523
7.26.2.12 VTSS_DIV64 . . . . .	523
7.26.2.13 VTSS_MOD64 . . . . .	523
7.26.2.14 VTSS_LABS . . . . .	523
7.26.2.15 VTSS_LLABS . . . . .	524
7.26.2.16 VTSS_OS_CTZ . . . . .	524
7.26.2.17 VTSS_OS_CTZ64 . . . . .	525
7.26.2.18 VTSS_OS_MALLOC . . . . .	525

7.26.2.19 VTSS_OS_FREE . . . . .	526
7.26.2.20 VTSS_OS_RAND . . . . .	526
7.27 vtss_api/include/vtss_otn_api.h File Reference . . . . .	526
7.27.1 Detailed Description . . . . .	526
7.28 vtss_api/include/vtss_packet_api.h File Reference . . . . .	526
7.28.1 Detailed Description . . . . .	529
7.28.2 Macro Definition Documentation . . . . .	529
7.28.2.1 VTSS_PRIO_SUPER . . . . .	530
7.28.2.2 VTSS_JR1_PACKET_HDR_SIZE_BYTES . . . . .	530
7.28.2.3 VTSS_JR2_PACKET_HDR_SIZE_BYTES . . . . .	530
7.28.2.4 VTSS_SVL_PACKET_HDR_SIZE_BYTES . . . . .	530
7.28.2.5 VTSS_L26_PACKET_HDR_SIZE_BYTES . . . . .	530
7.28.2.6 VTSS_PACKET_HDR_SIZE_BYTES . . . . .	531
7.28.2.7 VTSS_SVL_RX_IFH_SIZE . . . . .	531
7.28.2.8 VTSS_JR1_RX_IFH_SIZE . . . . .	531
7.28.2.9 VTSS_L26_RX_IFH_SIZE . . . . .	531
7.28.2.10 VTSS_JR2_RX_IFH_SIZE . . . . .	531
7.28.2.11 VTSS_PACKET_TX_IFH_MAX . . . . .	532
7.28.3 Enumeration Type Documentation . . . . .	532
7.28.3.1 vtss_packet_filter_t . . . . .	532
7.28.3.2 vtss_packet_oam_type_t . . . . .	532
7.28.3.3 vtss_packet_ptp_action_t . . . . .	533
7.28.3.4 vtss_tag_type_t . . . . .	533
7.28.3.5 vtss_packet_rx_hints_t . . . . .	533
7.28.3.6 vtss_packet_tx_vstax_t . . . . .	534
7.28.4 Function Documentation . . . . .	535
7.28.4.1 vtss_npi_conf_get() . . . . .	535
7.28.4.2 vtss_npi_conf_set() . . . . .	535
7.28.4.3 vtss_packet_rx_conf_get() . . . . .	536
7.28.4.4 vtss_packet_rx_conf_set() . . . . .	536

7.28.4.5 vtss_packet_rx_port_conf_get()	536
7.28.4.6 vtss_packet_rx_port_conf_set()	538
7.28.4.7 vtss_packet_rx_frame_get()	538
7.28.4.8 vtss_packet_rx_frame_get_raw()	539
7.28.4.9 vtss_packet_rx_frame_discard()	539
7.28.4.10 vtss_packet_tx_frame_port()	540
7.28.4.11 vtss_packet_tx_frame_port_vlan()	540
7.28.4.12 vtss_packet_tx_frame_vlan()	541
7.28.4.13 vtss_packet_frame_info_init()	541
7.28.4.14 vtss_packet_frame_filter()	541
7.28.4.15 vtss_packet_port_info_init()	542
7.28.4.16 vtss_packet_port_filter_get()	542
7.28.4.17 vtss_packet_rx_hdr_decode()	543
7.28.4.18 vtss_packet_tx_hdr_encode()	543
7.28.4.19 vtss_packet_tx_hdr_compile()	545
7.28.4.20 vtss_packet_tx_frame()	545
7.28.4.21 vtss_packet_tx_info_init()	546
7.28.4.22 vtss_packet_dma_conf_get()	546
7.28.4.23 vtss_packet_dma_conf_set()	547
7.28.4.24 vtss_packet_dma_offset()	547
7.29 vtss_api/include/vtss_pcs_10gbase_r_api.h File Reference	548
7.29.1 Detailed Description	548
7.30 vtss_api/include/vtss_phy_10g_api.h File Reference	548
7.30.1 Detailed Description	548
7.31 vtss_api/include/vtss_phy_api.h File Reference	548
7.31.1 Detailed Description	558
7.31.2 Macro Definition Documentation	558
7.31.2.1 MAX_CFG_BUF_SIZE	558
7.31.2.2 MAX_STAT_BUF_SIZE	558
7.31.2.3 VTSS_PHY_POWER_ACTIPHYS_BIT	559

7.31.2.4 VTSS_PHY_POWER_DYNAMIC_BIT . . . . .	559
7.31.2.5 VTSS_PHY_ACTIPHY_PWR . . . . .	559
7.31.2.6 VTSS_PHY_LINK_DOWN_PWR . . . . .	559
7.31.2.7 VTSS_PHY_LINK_UP_FULL_PWR . . . . .	559
7.31.2.8 VTSS_PHY_RECov_CLK1 . . . . .	560
7.31.2.9 VTSS_PHY_RECov_CLK2 . . . . .	560
7.31.2.10 VTSS_PHY_RECov_CLK_NUM . . . . .	560
7.31.2.11 VTSS_PHY_PAGE_STANDARD . . . . .	560
7.31.2.12 VTSS_PHY_PAGE_EXTENDED . . . . .	560
7.31.2.13 VTSS_PHY_PAGE_EXTENDED_2 . . . . .	561
7.31.2.14 VTSS_PHY_PAGE_EXTENDED_3 . . . . .	561
7.31.2.15 VTSS_PHY_PAGE_EXTENDED_4 . . . . .	561
7.31.2.16 VTSS_PHY_PAGE_GPIO . . . . .	561
7.31.2.17 VTSS_PHY_PAGE_1588 . . . . .	561
7.31.2.18 VTSS_PHY_PAGE_MACSEC . . . . .	562
7.31.2.19 VTSS_PHY_PAGE_TEST . . . . .	562
7.31.2.20 VTSS_PHY_PAGE_TR . . . . .	562
7.31.2.21 VTSS_PHY_PAGE_0x2DAF . . . . .	562
7.31.2.22 VTSS_PHY_REG_STANDARD . . . . .	562
7.31.2.23 VTSS_PHY_REG_EXTENDED . . . . .	563
7.31.2.24 VTSS_PHY_REG_GPIO . . . . .	563
7.31.2.25 VTSS_PHY_REG_TEST . . . . .	563
7.31.2.26 VTSS_PHY_REG_TR . . . . .	563
7.31.2.27 VTSS_PHY_LINK_LOS_EV . . . . .	563
7.31.2.28 VTSS_PHY_LINK_FFAIL_EV . . . . .	564
7.31.2.29 VTSS_PHY_LINK_AMS_EV . . . . .	564
7.31.2.30 VTSS_PHY_LINK_SPEED_STATE_CHANGE_EV . . . . .	564
7.31.2.31 VTSS_PHY_LINK_FDX_STATE_CHANGE_EV . . . . .	564
7.31.2.32 VTSS_PHY_LINK_AUTO_NEG_ERROR_EV . . . . .	564
7.31.2.33 VTSS_PHY_LINK_AUTO_NEG_COMPLETE_EV . . . . .	565

7.31.2.34 VTSS_PHY_LINK_INLINE_POW_DEV_DETECT_EV . . . . .	565
7.31.2.35 VTSS_PHY_LINK_SYMBOL_ERR_INT_EV . . . . .	565
7.31.2.36 VTSS_PHY_LINK_TX_FIFO_OVERFLOW_INT_EV . . . . .	565
7.31.2.37 VTSS_PHY_LINK_RX_FIFO_OVERFLOW_INT_EV . . . . .	565
7.31.2.38 VTSS_PHY_LINK_FALSE_CARRIER_INT_EV . . . . .	566
7.31.2.39 VTSS_PHY_LINK_LINK_SPEED_DS_DETECT_EV . . . . .	566
7.31.2.40 VTSS_PHY_LINK_MASTER_SLAVE_RES_ERR_EV . . . . .	566
7.31.2.41 VTSS_PHY_LINK_RX_ER_INT_EV . . . . .	566
7.31.2.42 VTSS_PHY_LINK_EXTENDED_REG_INT_EV . . . . .	566
7.31.2.43 VTSS_PHY_LINK_WAKE_ON_LAN_INT_EV . . . . .	567
7.31.2.44 VTSS_PHY_LINK_EXT_EEE_WAKE_ERR_EV . . . . .	567
7.31.2.45 VTSS_PHY_LINK_EXT_EEE_WAIT_TS_EV . . . . .	567
7.31.2.46 VTSS_PHY_LINK_EXT_EEE_WAIT_RX_TQ_EV . . . . .	567
7.31.2.47 VTSS_PHY_LINK_EXT_EEE_LINKFAIL_EV . . . . .	567
7.31.2.48 VTSS_PHY_LINK_EXT_RR_SW_COMPL_EV . . . . .	568
7.31.2.49 VTSS_PHY_LINK_EXT_MACSEC_HOST_MAC_EV . . . . .	568
7.31.2.50 VTSS_PHY_LINK_EXT_MACSEC_LINE_MAC_EV . . . . .	568
7.31.2.51 VTSS_PHY_LINK_EXT_MACSEC_FC_BUFF_EV . . . . .	568
7.31.2.52 VTSS_PHY_LINK_EXT_MACSEC_INGRESS_EV . . . . .	568
7.31.2.53 VTSS_PHY_LINK_EXT_MACSEC_EGRESS_EV . . . . .	569
7.31.2.54 VTSS_PHY_LINK_EXT_MEM_INT_RING_EV . . . . .	569
7.31.2.55 MAX_WOL_MAC_ADDR_SIZE . . . . .	569
7.31.2.56 MAX_WOL_PASSWD_SIZE . . . . .	569
7.31.2.57 MIN_WOL_PASSWD_SIZE . . . . .	569
7.31.3 Typedef Documentation . . . . .	570
7.31.3.1 vtss_phy_recov_clk_t . . . . .	570
7.31.3.2 vtss_phy_led_intensity . . . . .	570
7.31.4 Enumeration Type Documentation . . . . .	570
7.31.4.1 vtss_phy_led_mode_t . . . . .	570
7.31.4.2 vtss_phy_media_interface_t . . . . .	571

---

7.31.4.3 vtss_phy_mdi_t . . . . .	572
7.31.4.4 rgmii_skew_delay_psec_t . . . . .	572
7.31.4.5 vtss_phy_forced_reset_t . . . . .	573
7.31.4.6 vtss_phy_pkt_mode_t . . . . .	573
7.31.4.7 vtss_phy_mode_t . . . . .	573
7.31.4.8 vtss_phy_fast_link_fail_t . . . . .	574
7.31.4.9 vtss_phy_sigdet_polarity_t . . . . .	574
7.31.4.10 vtss_phy_unidirectional_t . . . . .	574
7.31.4.11 vtss_phy_mac_serdes_pcs_sgmii_pre . . . . .	574
7.31.4.12 vtss_phy_media_rem_fault_t . . . . .	576
7.31.4.13 vtss_phy_media_force_ams_sel_t . . . . .	576
7.31.4.14 vtss_phy_clk_source_t . . . . .	577
7.31.4.15 vtss_phy_freq_t . . . . .	577
7.31.4.16 vtss_phy_clk_squelch . . . . .	577
7.31.4.17 vtss_phy_gpio_mode_t . . . . .	578
7.31.4.18 lb_type . . . . .	578
7.31.4.19 vtss_wol_passwd_len_type_t . . . . .	579
7.31.4.20 vtss_fefi_mode_t . . . . .	579
7.31.5 Function Documentation . . . . .	579
7.31.5.1 vtss_phy_pre_reset() . . . . .	579
7.31.5.2 vtss_phy_post_reset() . . . . .	580
7.31.5.3 vtss_phy_pre_system_reset() . . . . .	580
7.31.5.4 vtss_phy_reset() . . . . .	581
7.31.5.5 vtss_phy_reset_get() . . . . .	581
7.31.5.6 vtss_phy_chip_temp_get() . . . . .	581
7.31.5.7 vtss_phy_chip_temp_init() . . . . .	582
7.31.5.8 vtss_phy_conf_get() . . . . .	582
7.31.5.9 vtss_phy_conf_set() . . . . .	583
7.31.5.10 vtss_phy_status_get() . . . . .	583
7.31.5.11 vtss_phy_cl37_lp_abil_get() . . . . .	584

7.31.5.12 vtss_phy_id_get()	584
7.31.5.13 vtss_phy_conf_1g_get()	584
7.31.5.14 vtss_phy_conf_1g_set()	585
7.31.5.15 vtss_phy_status_1g_get()	585
7.31.5.16 vtss_phy_power_conf_get()	586
7.31.5.17 vtss_phy_power_conf_set()	586
7.31.5.18 vtss_phy_power_status_get()	587
7.31.5.19 vtss_phy_clock_conf_set()	587
7.31.5.20 vtss_phy_clock_conf_get()	587
7.31.5.21 vtss_phy_i2c_read()	588
7.31.5.22 vtss_phy_i2c_write()	589
7.31.5.23 vtss_phy_read()	589
7.31.5.24 vtss_phy_read_page()	590
7.31.5.25 vtss_phy_mmd_read()	590
7.31.5.26 vtss_phy_mmd_write()	591
7.31.5.27 vtss_phy_write()	591
7.31.5.28 vtss_phy_write_masked()	592
7.31.5.29 vtss_phy_write_masked_page()	592
7.31.5.30 vtss_phy_gpio_mode()	593
7.31.5.31 vtss_phy_gpio_get()	593
7.31.5.32 vtss_phy_gpio_set()	594
7.31.5.33 vtss_phy_veriphy_start()	594
7.31.5.34 vtss_phy_veriphy_get()	595
7.31.5.35 vtss_phy_led_mode_set()	595
7.31.5.36 vtss_phy_led_intensity_set()	596
7.31.5.37 vtss_phy_led_intensity_get()	596
7.31.5.38 vtss_phy_enhanced_led_control_init()	596
7.31.5.39 vtss_phy_enhanced_led_control_init_get()	597
7.31.5.40 vtss_phy_coma_mode_disable()	597
7.31.5.41 vtss_phy_coma_mode_enable()	598

7.31.5.42 vga_adc_debug()	598
7.31.5.43 vtss_phy_port_eee_capable()	598
7.31.5.44 vtss_phy_eee_ena()	599
7.31.5.45 vtss_phy_eee_conf_get()	599
7.31.5.46 vtss_phy_eee_conf_set()	600
7.31.5.47 vtss_phy_eee_power_save_state_get()	600
7.31.5.48 vtss_phy_eee_link_partner_advertisements_get()	601
7.31.5.49 vtss_phy_event_enable_set()	601
7.31.5.50 vtss_phy_event_enable_get()	602
7.31.5.51 vtss_phy_event_poll()	602
7.31.5.52 vtss_squelch_workaround()	602
7.31.5.53 vtss_phy_csr_wr()	603
7.31.5.54 vtss_phy_csr_rd()	603
7.31.5.55 vtss_phy_statistic_get()	605
7.31.5.56 vtss_phy_do_page_chk_set()	605
7.31.5.57 vtss_phy_do_page_chk_get()	606
7.31.5.58 vtss_phy_loopback_set()	606
7.31.5.59 vtss_phy_loopback_get()	606
7.31.5.60 vtss_phy_is_8051_crc_ok()	607
7.31.5.61 vtss_phy_cfg_ob_post0()	607
7.31.5.62 vtss_phy_cfg_ib_cterm()	608
7.31.5.63 vtss_phy_atom12_patch_settings_get()	608
7.31.5.64 vtss_phy_reg_decode_status()	609
7.31.5.65 vtss_phy_flowcontrol_decode_status()	609
7.31.5.66 vtss_phy_debug_stat_print()	610
7.31.5.67 vtss_phy_warm_start_failed_get()	610
7.31.5.68 vtss_phy_debug_phyinfo_print()	611
7.31.5.69 vtss_phy_debug_register_dump()	611
7.31.5.70 vtss_phy_detect_base_ports()	612
7.31.5.71 vtss_phy_ext_connector_loopback()	612

7.31.5.72 vtss_phy_serdes_sgmii_loopback()	612
7.31.5.73 vtss_phy_serdes_fmedia_loopback()	613
7.31.5.74 vtss_phy_debug_regdump_print()	613
7.31.5.75 vtss_phy_wol_enable()	614
7.31.5.76 vtss_phy_wol_conf_get()	614
7.31.5.77 vtss_phy_wol_conf_set()	615
7.31.5.78 vtss_phy_patch_settings_get()	615
7.31.5.79 vtss_phy_serdes6g_rcpll_status_get()	616
7.31.5.80 vtss_phy_serdes1g_rcpll_status_get()	616
7.31.5.81 vtss_phy_lcpll_status_get()	617
7.31.5.82 vtss_phy_reset_lcpll()	617
7.31.5.83 vtss_phy_sd6g_ob_post_rd()	618
7.31.5.84 vtss_phy_sd6g_ob_post_wr()	618
7.31.5.85 vtss_phy_sd6g_ob_lev_rd()	619
7.31.5.86 vtss_phy_sd6g_ob_lev_wr()	619
7.31.5.87 vtss_phy_mac_media_inhibit_odd_start()	620
7.31.5.88 vtss_phy_fefi_get()	620
7.31.5.89 vtss_phy_fefi_set()	620
7.31.5.90 vtss_phy_fefi_detect()	621
7.31.5.91 vtss_phy_mse_100m_get()	621
7.31.5.92 vtss_phy_mse_1000m_get()	622
7.31.5.93 vtss_phy_read_tr_addr()	622
7.31.5.94 vtss_phy_is_viper_revB()	623
7.31.5.95 vtss_phy_ext_event_poll()	623
7.31.5.96 vtss_phy_status_inst_poll()	624
7.31.5.97 vtss_phy_macsec_csr_sd6g_rd()	624
7.31.5.98 vtss_phy_macsec_csr_sd6g_wr()	625
7.31.5.99 vtss_phy_sd6g_mac_serdes_conf()	625
7.32 vtss_api/include/vtss_phy_ts_api.h File Reference	625
7.32.1 Detailed Description	626

---

7.33 vtss_api/include/vtss_port_api.h File Reference . . . . .	626
7.33.1 Detailed Description . . . . .	628
7.33.2 Macro Definition Documentation . . . . .	628
7.33.2.1 CHIP_PORT_UNUSED . . . . .	628
7.33.2.2 VTSS_FRAME_GAP_DEFAULT . . . . .	628
7.33.2.3 VTSS_MAX_FRAME_LENGTH_STANDARD . . . . .	629
7.33.2.4 VTSS_MAX_FRAME_LENGTH_MAX [1/2] . . . . .	629
7.33.2.5 VTSS_MAX_FRAME_LENGTH_MAX [2/2] . . . . .	629
7.33.3 Enumeration Type Documentation . . . . .	629
7.33.3.1 vtss_miim_controller_t . . . . .	629
7.33.3.2 vtss_internal_bw_t . . . . .	630
7.33.3.3 vtss_port_clause_37_remote_fault_t . . . . .	630
7.33.3.4 vtss_port_max_tags_t . . . . .	630
7.33.3.5 vtss_port_loop_t . . . . .	631
7.33.3.6 vtss_port_forward_t . . . . .	631
7.33.4 Function Documentation . . . . .	631
7.33.4.1 vtss_port_map_set() . . . . .	631
7.33.4.2 vtss_port_map_get() . . . . .	632
7.33.4.3 vtss_port_clause_37_control_get() . . . . .	632
7.33.4.4 vtss_port_clause_37_control_set() . . . . .	633
7.33.4.5 vtss_port_conf_set() . . . . .	633
7.33.4.6 vtss_port_conf_get() . . . . .	634
7.33.4.7 vtss_port_status_get() . . . . .	634
7.33.4.8 vtss_port_counters_update() . . . . .	634
7.33.4.9 vtss_port_counters_clear() . . . . .	635
7.33.4.10 vtss_port_counters_get() . . . . .	635
7.33.4.11 vtss_port_basic_counters_get() . . . . .	636
7.33.4.12 vtss_port_forward_state_get() . . . . .	636
7.33.4.13 vtss_port_forward_state_set() . . . . .	637
7.33.4.14 vtss_miim_read() . . . . .	637

7.33.4.15 <a href="#">vtss_miim_write()</a>	638
7.34 <a href="#">vtss_api/include/vtss_qos_api.h</a> File Reference	638
7.34.1 Detailed Description	640
7.34.2 Macro Definition Documentation	640
7.34.2.1 <a href="#">VTSS_PORT_POLICERS</a>	640
7.34.2.2 <a href="#">VTSS_PORT_POLICER_CPU_QUEUES</a>	641
7.34.2.3 <a href="#">VTSS_QCL_IDS</a>	641
7.34.2.4 <a href="#">VTSS_QCL_ID_START</a>	641
7.34.2.5 <a href="#">VTSS_QCL_ID_END</a>	641
7.34.2.6 <a href="#">VTSS_QCL_ARRAY_SIZE</a>	641
7.34.2.7 <a href="#">VTSS_QCE_ID_LAST</a>	642
7.34.3 Enumeration Type Documentation	642
7.34.3.1 <a href="#">vtss_tag_remark_mode_t</a>	642
7.34.3.2 <a href="#">vtss_dscp_mode_t</a>	642
7.34.3.3 <a href="#">vtss_dscp_emode_t</a>	642
7.34.3.4 <a href="#">vtss_qce_type_t</a>	643
7.34.4 Function Documentation	643
7.34.4.1 <a href="#">vtss_qos_conf_get()</a>	643
7.34.4.2 <a href="#">vtss_qos_conf_set()</a>	644
7.34.4.3 <a href="#">vtss_mep_policer_conf_get()</a>	644
7.34.4.4 <a href="#">vtss_mep_policer_conf_set()</a>	645
7.34.4.5 <a href="#">vtss_qos_port_conf_get()</a>	645
7.34.4.6 <a href="#">vtss_qos_port_conf_set()</a>	645
7.34.4.7 <a href="#">vtss_qce_init()</a>	646
7.34.4.8 <a href="#">vtss_qce_add()</a>	646
7.34.4.9 <a href="#">vtss_qce_del()</a>	647
7.35 <a href="#">vtss_api/include/vtss_rab_api.h</a> File Reference	647
7.35.1 Detailed Description	647
7.36 <a href="#">vtss_api/include/vtss_security_api.h</a> File Reference	647
7.36.1 Detailed Description	650

---

7.36.2 Macro Definition Documentation . . . . .	650
7.36.2.1 VTSS_ACE_ID_LAST . . . . .	650
7.36.3 Enumeration Type Documentation . . . . .	650
7.36.3.1 vtss_auth_state_t . . . . .	650
7.36.3.2 vtss_acl_port_action_t . . . . .	651
7.36.3.3 vtss_acl_ptp_action_t . . . . .	651
7.36.3.4 vtss_ace_type_t . . . . .	651
7.36.3.5 vtss_ace_bit_t . . . . .	652
7.36.4 Function Documentation . . . . .	652
7.36.4.1 vtss_auth_port_state_get() . . . . .	652
7.36.4.2 vtss_auth_port_state_set() . . . . .	652
7.36.4.3 vtss_acl_policer_conf_get() . . . . .	653
7.36.4.4 vtss_acl_policer_conf_set() . . . . .	653
7.36.4.5 vtss_acl_port_conf_get() . . . . .	654
7.36.4.6 vtss_acl_port_conf_set() . . . . .	654
7.36.4.7 vtss_acl_port_counter_get() . . . . .	655
7.36.4.8 vtss_acl_port_counter_clear() . . . . .	655
7.36.4.9 vtss_ace_init() . . . . .	655
7.36.4.10 vtss_ace_add() . . . . .	656
7.36.4.11 vtss_ace_del() . . . . .	656
7.36.4.12 vtss_ace_counter_get() . . . . .	657
7.36.4.13 vtss_ace_counter_clear() . . . . .	657
7.37 vtss_api/include/vtss_sfi4_api.h File Reference . . . . .	657
7.37.1 Detailed Description . . . . .	658
7.38 vtss_api/include/vtss_sync_api.h File Reference . . . . .	658
7.38.1 Detailed Description . . . . .	659
7.38.2 Macro Definition Documentation . . . . .	659
7.38.2.1 VTSS_SYNCE_CLK_A . . . . .	659
7.38.2.2 VTSS_SYNCE_CLK_B . . . . .	659
7.38.2.3 VTSS_SYNCE_CLK_MAX . . . . .	659

7.38.3 Enumeration Type Documentation . . . . .	659
7.38.3.1 vtss_syncce_divider_t . . . . .	659
7.38.4 Function Documentation . . . . .	660
7.38.4.1 vtss_syncce_clock_out_set() . . . . .	660
7.38.4.2 vtss_syncce_clock_out_get() . . . . .	660
7.38.4.3 vtss_syncce_clock_in_set() . . . . .	661
7.38.4.4 vtss_syncce_clock_in_get() . . . . .	661
7.39 vtss_api/include/vtss_tfi5_api.h File Reference . . . . .	661
7.39.1 Detailed Description . . . . .	662
7.40 vtss_api/include/vtss_ts_api.h File Reference . . . . .	662
7.40.1 Detailed Description . . . . .	665
7.40.2 Function Documentation . . . . .	665
7.40.2.1 vtss_ts_timeofday_set() . . . . .	665
7.40.2.2 vtss_ts_timeofday_set_delta() . . . . .	665
7.40.2.3 vtss_ts_timeofday_offset_set() . . . . .	666
7.40.2.4 vtss_ts_adjtimer_one_sec() . . . . .	666
7.40.2.5 vtss_ts_ongoing_adjustment() . . . . .	667
7.40.2.6 vtss_ts_timeofday_get() . . . . .	667
7.40.2.7 vtss_ts_timeofday_next_pps_get() . . . . .	667
7.40.2.8 vtss_ts_adjtimer_set() . . . . .	668
7.40.2.9 vtss_ts_adjtimer_get() . . . . .	668
7.40.2.10 vtss_ts_freq_offset_get() . . . . .	669
7.40.2.11 vtss_ts_external_clock_mode_get() . . . . .	669
7.40.2.12 vtss_ts_external_clock_mode_set() . . . . .	669
7.40.2.13 vtss_ts_external_clock_saved_get() . . . . .	670
7.40.2.14 vtss_ts_ingress_latency_set() . . . . .	670
7.40.2.15 vtss_ts_ingress_latency_get() . . . . .	671
7.40.2.16 vtss_ts_p2p_delay_set() . . . . .	671
7.40.2.17 vtss_ts_p2p_delay_get() . . . . .	671
7.40.2.18 vtss_ts_egress_latency_set() . . . . .	672

7.40.2.19 <code>vtss_ts_egress_latency_get()</code>	672
7.40.2.20 <code>vtss_ts_delay_asymmetry_set()</code>	673
7.40.2.21 <code>vtss_ts_delay_asymmetry_get()</code>	673
7.40.2.22 <code>vtss_ts_operation_mode_set()</code>	674
7.40.2.23 <code>vtss_ts_operation_mode_get()</code>	674
7.40.2.24 <code>vtss_ts_internal_mode_set()</code>	674
7.40.2.25 <code>vtss_ts_internal_mode_get()</code>	675
7.40.2.26 <code>vtss_tx_timestamp_update()</code>	675
7.40.2.27 <code>vtss_rx_timestamp_get()</code>	676
7.40.2.28 <code>vtss_rx_timestamp_id_release()</code>	676
7.40.2.29 <code>vtss_rx_master_timestamp_get()</code>	676
7.40.2.30 <code>vtss_tx_timestamp_idx_alloc()</code>	677
7.40.2.31 <code>vtss_timestamp_age()</code>	677
7.40.2.32 <code>vtss_ts_status_change()</code>	678
7.41 <code>vtss_api/include/vtss_upi_api.h</code> File Reference	678
7.41.1 Detailed Description	678
7.42 <code>vtss_api/include/vtss_wis_api.h</code> File Reference	678
7.42.1 Detailed Description	678
7.43 <code>vtss_api/include/vtss_xaui_api.h</code> File Reference	678
7.43.1 Detailed Description	679
7.44 <code>vtss_api/include/vtss_xfi_api.h</code> File Reference	679
7.44.1 Detailed Description	679
<b>Index</b>	<b>681</b>



# Chapter 1

## Ethernet Virtual Connections

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

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

### 1.1 Port Configuration

The following EVC functions are available per port.

- `vtss_evc_port_conf_get()` is used to get the EVC port configuration.
- `vtss_evc_port_conf_set()` is used to set the EVC port configuration.

### 1.2 Policer Configuration

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

- `vtss_evc_policer_conf_get()` is used to get the EVC policer configuration.
- `vtss_evc_policer_conf_set()` is used to set the EVC policer configuration.

By default, all EVC policers are disabled.

## 1.3 EVCs

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

- `vtss_evc_add()` is used to add an EVC rule.
- `vtss_evc_del()` is used to delete an EVC rule.
- `vtss_evc_get()` is used to get an EVC rule.

The `vtss_evc_conf_t` structure used when adding an EVC rule includes the following:

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

By default, no EVCs are setup.

## 1.4 ECEs

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

- `vtss_ece_add()` is used to add an ECE.
- `vtss_ece_del()` is used to delete an ECE.

The `vtss_ece_t` structure used when adding an ECE includes the following fields:

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

By default, no ECEs are setup.

## 1.5 EVC Port Statistics

EVC statistics include green/yellow/red/discardable frames. These counters are available per port and priority.

- `vtss_port_counters_get()` is used to get port counters.
- `vtss_port_counters_clear()` is used to clear port counters.

# Chapter 2

## Layer 2

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

### 2.1 MAC Address Table

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

- [vtss\\_mac\\_table\\_add\(\)](#) is used to add a static entry.
- [vtss\\_mac\\_table\\_del\(\)](#) is used to delete a static entry.
- [vtss\\_mac\\_table\\_get\(\)](#) is used to lookup a specific entry.
- [vtss\\_mac\\_table\\_get\\_next\(\)](#) is used to get the next entry for table traversal.
- [vtss\\_mac\\_table\\_age\\_time\\_get\(\)](#) is used to get the age time.
- [vtss\\_mac\\_table\\_age\\_time\\_set\(\)](#) is used to set the age time.
- [vtss\\_mac\\_table\\_age\(\)](#) is used for manual age scan.
- [vtss\\_mac\\_table\\_vlan\\_age\(\)](#) is used for manual age scan per VLAN.
- [vtss\\_mac\\_table\\_flush\(\)](#) is used to flush all dynamic entries.
- [vtss\\_mac\\_table\\_port\\_flush\(\)](#) is used to flush dynamic entries per port.
- [vtss\\_mac\\_table\\_vlan\\_flush\(\)](#) is used to flush dynamic entries per VLAN.
- [vtss\\_mac\\_table\\_vlan\\_port\\_flush\(\)](#) is used to flush dynamic entries per VLAN and port.
- [vtss\\_mac\\_table\\_status\\_get\(\)](#) is used to poll for MAC address table change events.
- [vtss\\_learn\\_port\\_mode\\_get\(\)](#) is used to get the learn mode per port.
- [vtss\\_learn\\_port\\_mode\\_set\(\)](#) is used to set the learn mode per port.

By default, automatic learning and ageing is enabled.

## 2.2 Operational State

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

- [vtss\\_port\\_state\\_get\(\)](#) is used to get the forwarding state for each port.
- [vtss\\_port\\_state\\_set\(\)](#) is used to set the forwarding state for each port.

By default, all ports are down.

## 2.3 Spanning Tree

The following Spanning Tree functions are available:

- [vtss\\_stp\\_port\\_state\\_get\(\)](#) is used to get the STP state for a port.
- [vtss\\_stp\\_port\\_state\\_set\(\)](#) is used to set the STP state for a port.
- [vtss\\_mstp\\_vlan\\_msti\\_get\(\)](#) is used to get the mapping from VLAN to MSTP instance.
- [vtss\\_mstp\\_vlan\\_msti\\_set\(\)](#) is used to set the mapping from VLAN to MSTP instance.
- [vtss\\_mstp\\_port\\_msti\\_state\\_get\(\)](#) is used to get the MSTP state per MSTP instance and port.
- [vtss\\_mstp\\_port\\_msti\\_state\\_set\(\)](#) is used to set the MSTP state per MSTP instance and port.

By default, all ports are in STP forwarding mode.

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

## 2.4 VLAN

Basic VLAN classification and tag preservation works as follows:

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

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

The following VLAN functions are available:

- [vtss\\_vlan\\_conf\\_get\(\)](#) is used to get the global VLAN configuration.

- [vtss\\_vlan\\_conf\\_set\(\)](#) is used to set the global VLAN configuration.
- [vtss\\_vlan\\_port\\_conf\\_get\(\)](#) is used to get the VLAN port configuration.
- [vtss\\_vlan\\_port\\_conf\\_set\(\)](#) is used to set the VLAN port configuration.
- [vtss\\_vlan\\_tx\\_tag\\_get\(\)](#) is used to get the advanced tagging configuration for a VLAN.
- [vtss\\_vlan\\_tx\\_tag\\_set\(\)](#) is used to set the advanced tagging configuration for a VLAN.
- [vtss\\_vlan\\_port\\_members\\_get\(\)](#) is used to get the VLAN port members.
- [vtss\\_vlan\\_port\\_members\\_set\(\)](#) is used to set the VLAN port members.
- [vtss\\_vlan\\_vid\\_conf\\_get\(\)](#) is used to get VLAN configuration.
- [vtss\\_vlan\\_vid\\_conf\\_set\(\)](#) is used to set VLAN configuration.

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

## 2.5 VLAN Classification List

Advanced VLAN classification rules can be done using the VLAN Classification List (VCL). Each VLAN Classification Entry (VCE) is identified by a VCE ID ([vtss\\_vce\\_id\\_t](#)). The following VCL functions are available:

- [vtss\\_vcl\\_port\\_conf\\_get\(\)](#) is used to get the VCL port configuration.
- [vtss\\_vcl\\_port\\_conf\\_set\(\)](#) is used to set the VCL port configuration.
- [vtss\\_vce\\_init\(\)](#) is used to initialize a VCE to default values.
- [vtss\\_vce\\_add\(\)](#) is used to add or modify a VCE.
- [vtss\\_vce\\_del\(\)](#) is used to delete a VCE.

The VCEs are ordered in a list of rules based on the VCE IDs. When adding a rule, the VCE ID of the rule and the VCE ID of the next rule in the list must be specified. A special value [VTSS\\_VCE\\_ID\\_LAST](#) is used to specify that the rule must be added at the end of the list.

Each VCE includes a key structure ([vtss\\_vce\\_key\\_t](#)) with fields used for matching received frames and an action structure ([vtss\\_vce\\_action\\_t](#)) with the classified VLAN ID.

By default, no VCE rules are setup.

## 2.6 VLAN Translation

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

- [vtss\\_vlan\\_trans\\_group\\_to\\_port\\_get\(\)](#) is used to get the ports of a translation group.
- [vtss\\_vlan\\_trans\\_group\\_to\\_port\\_set\(\)](#) is used to set the ports of a translation group.
- [vtss\\_vlan\\_trans\\_group\\_add\(\)](#) is used to add a VLAN translation to a group.
- [vtss\\_vlan\\_trans\\_group\\_del\(\)](#) is used to delete a VLAN translation from a group.

By default, no VLAN translation rules are setup.

## 2.7 Port Isolation

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

- [vtss\\_isolated\\_vlan\\_get\(\)](#) is used to get the isolation mode for a VLAN.
- [vtss\\_isolated\\_vlan\\_set\(\)](#) is used to set the isolation mode for a VLAN.
- [vtss\\_isolated\\_port\\_members\\_get\(\)](#) is used to get the isolated port members.
- [vtss\\_isolated\\_port\\_members\\_set\(\)](#) is used to get the isolated port members.

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

## 2.8 Private VLAN

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

- [vtss\\_pvlan\\_port\\_members\\_get\(\)](#) is used to get the port members of PVLAN.
- [vtss\\_pvlan\\_port\\_members\\_set\(\)](#) is used to set the port members of PVLAN.

By default, all ports are included in PVLAN 1.

## 2.9 Asymmetric Private VLAN

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

- [vtss\\_apvlan\\_port\\_members\\_get\(\)](#) is used to get the egress port members for an ingress port.
- [vtss\\_apvlan\\_port\\_members\\_set\(\)](#) is used to set the egress port members for an ingress port.

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

## 2.10 Destination Port Groups

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

- [vtss\\_dgroup\\_port\\_conf\\_get\(\)](#) is used to get the destination group for a port.
- [vtss\\_dgroup\\_port\\_conf\\_set\(\)](#) is used to set the destination group for a port.

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

## 2.11 sFlow

The sFlow functions can be used to sample frame flows.

- [vtss\\_sflow\\_port\\_conf\\_get\(\)](#) is used to get the sFlow port configuration.
- [vtss\\_sflow\\_port\\_conf\\_set\(\)](#) is used to set the sFlow port configuration.
- [vtss\\_sflow\\_sampling\\_rate\\_convert\(\)](#) converts desired sampling rate to actual sampling rate.

By default, sFlow is disabled on all ports.

## 2.12 Link Aggregation

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

- [vtss\\_aggr\\_port\\_members\\_get\(\)](#) is used to get the aggregation port members.
- [vtss\\_aggr\\_port\\_members\\_set\(\)](#) is used to set the aggregation port members.
- [vtss\\_aggr\\_mode\\_get\(\)](#) is used to get the aggregation mode.
- [vtss\\_aggr\\_mode\\_set\(\)](#) is used to set the aggregation mode.

By default, no link aggregations exist.

## 2.13 Mirroring

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

- [vtss\\_mirror\\_conf\\_get\(\)](#) is used to get the mirror configuration.
- [vtss\\_mirror\\_conf\\_set\(\)](#) is used to set the mirror configuration.
- [vtss\\_mirror\\_monitor\\_port\\_get\(\)](#) is used to get the mirror monitor port.
- [vtss\\_mirror\\_monitor\\_port\\_set\(\)](#) is used to set the mirror monitor port.
- [vtss\\_mirror\\_ingress\\_ports\\_get\(\)](#) is used to get the ingress mirroring port members.
- [vtss\\_mirror\\_ingress\\_ports\\_set\(\)](#) is used to set the ingress mirroring port members.
- [vtss\\_mirror\\_egress\\_ports\\_get\(\)](#) is used to get the egress mirroring port members.
- [vtss\\_mirror\\_egress\\_ports\\_set\(\)](#) is used to set the egress mirroring port members.
- [vtss\\_mirror\\_cpu\\_ingress\\_get\(\)](#) is used to get the CPU ingress mirror mode.
- [vtss\\_mirror\\_cpu\\_ingress\\_set\(\)](#) is used to set the CPU ingress mirror mode.
- [vtss\\_mirror\\_cpu\\_egress\\_get\(\)](#) is used to get the CPU egress mirror mode.
- [vtss\\_mirror\\_cpu\\_egress\\_set\(\)](#) is used to set the CPU egress mirror mode.

By default, mirroring is disabled for all ports.

## 2.14 Flooding Control

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

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

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

## 2.15 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.
- `vtss_ipv4_mc_add()` is used to add a source specific IPv4 multicast entry.
- `vtss_ipv4_mc_del()` is used to delete a source specific IPv4 multicast entry.

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

## 2.16 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.
- `vtss_ipv6_mc_add()` is used to add a source specific IPv6 multicast entry.
- `vtss_ipv6_mc_del()` is used to delete a source specific IPv6 multicast entry.

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

## 2.17 Ethernet Protection Switching

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

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

By default, all ports are unprotected.

## 2.18 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.



# Chapter 3

## Security

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

### 3.1 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.

### 3.2 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.

#### 3.2.1 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

### 3.2.2 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.

### 3.2.3 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 4

# Data Structure Index

### 4.1 Data Structures

Here are the data structures with brief descriptions:

<code>port_custom_conf_t</code>	Port configuration . . . . .	23
<code>serdes_fields_t</code>	Serdes fields . . . . .	26
<code>vtss_ace_frame_arp_t</code>	Frame data for VTSS_ACE_TYPE_ARP . . . . .	27
<code>vtss_ace_frame_etype_t</code>	Frame data for VTSS_ACE_TYPEETYPE . . . . .	30
<code>vtss_ace_frame_ipv4_t</code>	Frame data for VTSS_ACE_TYPE_IPV4 . . . . .	32
<code>vtss_ace_frame_ipv6_t</code>	Frame data for VTSS_ACE_TYPE_IPV6 . . . . .	37
<code>vtss_ace_frame_llc_t</code>	Frame data for VTSS_ACE_TYPE_LLCC . . . . .	41
<code>vtss_ace_frame_snap_t</code>	Frame data for VTSS_ACE_TYPE_SNAP . . . . .	42
<code>vtss_ace_ptp_t</code>	PTP header filtering . . . . .	43
<code>vtss_ace_sip_smac_t</code>	SIP/SMAC filtering . . . . .	44
<code>vtss_ace_t</code>	Access Control Entry . . . . .	45
<code>vtss_ace_vlan_t</code>	ACE VLAN information . . . . .	49
<code>vtss_acl_action_t</code>	ACL Action . . . . .	51
<code>vtss_acl_policer_conf_t</code>	ACL policer configuration . . . . .	54
<code>vtss_acl_port_conf_t</code>	ACL port configuration . . . . .	55
<code>vtss_aggr_mode_t</code>	Aggregation traffic distribution mode . . . . .	56
<code>vtss_aneg_t</code>	Auto negotiation struct . . . . .	57
<code>vtss_api_lock_t</code>	API lock structure . . . . .	58

<code>vtss_basic_counters_t</code>	Basic counters structure . . . . .	60
<code>vtss_chip_id_t</code>	Chip ID . . . . .	60
<code>vtss_counter_pair_t</code>	Counter pair . . . . .	61
<code>vtss_debug_info_t</code>	Debug information structure . . . . .	62
<code>vtss_debug_lock_t</code>	API debug lock structure . . . . .	64
<code>vtss_dgroup_port_conf_t</code>	Destination group port configuration . . . . .	65
<code>vtss_dlb_policer_conf_t</code>	Dual leaky buckets policer configuration . . . . .	66
<code>vtss_ece_action_t</code>	ECE action . . . . .	68
<code>vtss_ece_frame_ipv4_t</code>	ECE IPv4 information . . . . .	70
<code>vtss_ece_frame_ipv6_t</code>	ECE IPv6 information . . . . .	72
<code>vtss_ece_key_t</code>	ECE key . . . . .	73
<code>vtss_ece_mac_t</code>	ECE MAC information . . . . .	75
<code>vtss_ece_outer_tag_t</code>	ECE outer tag . . . . .	76
<code>vtss_ece_t</code>	EVC Control Entry . . . . .	78
<code>vtss_ece_tag_t</code>	ECE tag information . . . . .	79
<code>vtss_eee_port_conf_t</code>	EEE port configuration . . . . .	80
<code>vtss_eee_port_counter_t</code>	EEE port counters (JR only) . . . . .	82
<code>vtss_eee_port_state_t</code>	EEE port state (JR only) . . . . .	84
<code>vtss_eps_port_conf_t</code>	Port protection configuration . . . . .	84
<code>vtss_evc_conf_t</code>	EVC configuration (excluding UNIs) . . . . .	85
<code>vtss_evc_inner_tag_t</code>	EVC inner tag . . . . .	87
<code>vtss_evc_pb_conf_t</code>	PB specific EVC configuration . . . . .	88
<code>vtss_evc_port_conf_t</code>	EVC port configuration . . . . .	90
<code>vtss_fan_conf_t</code>	Fan specifications . . . . .	91
<code>vtss_init_conf_t</code>	Initialization configuration . . . . .	93
<code>vtss_inst_create_t</code>	Create structure . . . . .	97
<code>vtss_intr_t</code>	Interrupt source structure . . . . .	98
<code>vtss_ip_addr_t</code>	Either an IPv4 or IPv6 address . . . . .	98
<code>vtss_ip_network_t</code>	IPv6 network . . . . .	100

vtss_ipv4_network_t	IPv4 network . . . . .	100
vtss_ipv4_uc_t	IPv4 unicast routing entry . . . . .	101
vtss_ipv6_network_t	IPv6 network . . . . .	102
vtss_ipv6_t	IPv6 address/mask . . . . .	103
vtss_ipv6_uc_t	IPv6 routing entry . . . . .	104
vtss_irq_conf_t	Interrupt configuration options . . . . .	105
vtss_irq_status_t	Interrupt status structure . . . . .	106
vtss_l3_counters_t	Routing interface statics counter . . . . .	107
vtss_lcpll_status_t	Structure for Get PHY LC-PLL status . . . . .	109
vtss_learn_mode_t	Learning mode . . . . .	111
vtss_mac_t	MAC Address . . . . .	112
vtss_mac_table_entry_t	MAC address entry . . . . .	113
vtss_mac_table_status_t	MAC address table status . . . . .	115
vtss_mce_action_t	MCE action . . . . .	116
vtss_mce_key_t	MCE key . . . . .	118
vtss_mce_t	MEP Control Entry . . . . .	119
vtss_mirror_conf_t	Mirror configuration . . . . .	120
vtss_mtimer_t	Timer structure . . . . .	121
vtss_npi_conf_t	NPI configuration . . . . .	122
vtss_os_timestamp_t	. . . . .	123
vtss_packet_dma_conf_t	. . . . .	124
vtss_packet_frame_info_t	Information about frame . . . . .	125
vtss_packet_port_filter_t	Packet information for each port . . . . .	126
vtss_packet_port_info_t	Port info structure . . . . .	127
vtss_packet_rx_conf_t	CPU Rx configuration . . . . .	129
vtss_packet_rx_header_t	System frame header describing received frame . . . . .	130
vtss_packet_rx_info_t	Decoded extraction header properties . . . . .	132
vtss_packet_rx_meta_t	Input structure to <code>vtss_packet_rx_hdr_decode()</code> . . . . .	141
vtss_packet_rx_port_conf_t	Packet registration per port . . . . .	145
vtss_packet_rx_queue_conf_t	CPU Rx queue configuration . . . . .	146

<a href="#">vtss_packet_rx_queue_map_t</a>	CPU Rx queue map . . . . .	147
<a href="#">vtss_packet_rx_queue_npi_conf_t</a>	CPU Rx queue NPI configuration . . . . .	149
<a href="#">vtss_packet_rx_reg_t</a>	CPU Rx packet registration . . . . .	150
<a href="#">vtss_packet_tx_ifh_t</a>	Compiled Tx Frame Header . . . . .	152
<a href="#">vtss_packet_tx_info_t</a>	Injection Properties . . . . .	153
<a href="#">vtss_phy_aneg_t</a>	PHY auto negotiation advertisement . . . . .	162
<a href="#">vtss_phy_clock_conf_t</a>	PHY clock configuration . . . . .	164
<a href="#">vtss_phy_conf_1g_t</a>	PHY 1G configuration . . . . .	166
<a href="#">vtss_phy_conf_t</a>	PHY configuration . . . . .	167
<a href="#">vtss_phy_eee_conf_t</a>	EEE configuration . . . . .	170
<a href="#">vtss_phy_enhanced_led_control_t</a>	Enhanced LED control . . . . .	170
<a href="#">vtss_phy_forced_t</a>	PHY forced mode configuration . . . . .	172
<a href="#">vtss_phy_led_mode_select_t</a>	LED model selection . . . . .	173
<a href="#">vtss_phy_loopback_t</a>	1G Phy loopbacks . . . . .	174
<a href="#">vtss_phy_mac_serdes_pcs_ctrl_t</a>	PHY MAC SerDes PCS Control, Reg16E3 . . . . .	176
<a href="#">vtss_phy_media_serdes_pcs_ctrl_t</a>	PHY MEDIA SerDes PCS Control, Reg23E3 . . . . .	179
<a href="#">vtss_phy_power_conf_t</a>	PHY power configuration . . . . .	182
<a href="#">vtss_phy_power_status_t</a>	PHY power status . . . . .	182
<a href="#">vtss_phy_reset_conf_t</a>	PHY reset structure . . . . .	183
<a href="#">vtss_phy_rgmii_conf_t</a>	PHY RGMII configuration . . . . .	185
<a href="#">vtss_phy_statistic_t</a>	Phy statistic information . . . . .	186
<a href="#">vtss_phy_status_1g_t</a>	PHY 1G status . . . . .	188
<a href="#">vtss_phy_tbi_conf_t</a>	PHY TBI configuration . . . . .	189
<a href="#">vtss_phy_type_t</a>	Phy type information . . . . .	190
<a href="#">vtss_phy_veriphy_result_t</a>	VeriPHY result . . . . .	192
<a href="#">vtss_phy_wol_conf_t</a>	Structure for Get/Set Wake-On-LAN configuration . . . . .	193
<a href="#">vtss_pi_conf_t</a>	PI configuration . . . . .	194
<a href="#">vtss_policer_ext_t</a>	Policer Extensions . . . . .	196
<a href="#">vtss_policer_t</a>	Policer . . . . .	196

<a href="#">vtss_port_bridge_counters_t</a>	Port bridge counter structure (RFC 4188) . . . . .	197
<a href="#">vtss_port_clause_37_adv_t</a>	Advertisement control data for Clause 37 aneg . . . . .	198
<a href="#">vtss_port_clause_37_control_t</a>	Auto-negotiation control parameter struct . . . . .	200
<a href="#">vtss_port_conf_t</a>	Port configuration structure . . . . .	201
<a href="#">vtss_port_counters_t</a>	Port counter structure . . . . .	205
<a href="#">vtss_port_ethernet_like_counters_t</a>	Ethernet-like Interface counter structure (RFC 3635) . . . . .	207
<a href="#">vtss_port_evc_counters_t</a>	EVC counters . . . . .	208
<a href="#">vtss_port_flow_control_conf_t</a>	Flow control setup . . . . .	210
<a href="#">vtss_port_frame_gaps_t</a>	Inter frame gap structure . . . . .	211
<a href="#">vtss_port_if_group_counters_t</a>	Interfaces Group counter structure (RFC 2863) . . . . .	213
<a href="#">vtss_port_map_t</a>	Port map structure . . . . .	216
<a href="#">vtss_port_proprietary_counters_t</a>	Port proprietary counter structure . . . . .	218
<a href="#">vtss_port_rmon_counters_t</a>	RMON counter structure (RFC 2819) . . . . .	218
<a href="#">vtss_port_serdes_conf_t</a>	SFI Serdes configuration . . . . .	225
<a href="#">vtss_port_sgmii_aneg_t</a>	Advertisement control data for SGMII aneg . . . . .	226
<a href="#">vtss_port_status_t</a>	Port status parameter struct . . . . .	228
<a href="#">vtss_qce_action_t</a>	QCE action . . . . .	231
<a href="#">vtss_qce_frame_etype_t</a>	Frame data for VTSS_QCE_TYPE_ETYPE . . . . .	234
<a href="#">vtss_qce_frame_ipv4_t</a>	Frame data for VTSS_QCE_TYPE_IPV4 . . . . .	234
<a href="#">vtss_qce_frame_ipv6_t</a>	Frame data for VTSS_QCE_TYPE_IPV6 . . . . .	236
<a href="#">vtss_qce_frame_llc_t</a>	Frame data for VTSS_QCE_TYPE_LLCC . . . . .	238
<a href="#">vtss_qce_frame_snap_t</a>	Frame data for VTSS_QCE_TYPE_SNAP . . . . .	239
<a href="#">vtss_qce_key_t</a>	QCE key . . . . .	239
<a href="#">vtss_qce_mac_t</a>	QCE MAC information . . . . .	242
<a href="#">vtss_qce_t</a>	QoS Control Entry . . . . .	243
<a href="#">vtss_qce_tag_t</a>	QCE tag information . . . . .	244
<a href="#">vtss_qos_conf_t</a>	All parameters below are defined per chip . . . . .	246
<a href="#">vtss_qos_port_conf_t</a>	QoS setup per port . . . . .	250
<a href="#">vtss_rcpll_status_t</a>	Structure for Get PHY RC-PLL status . . . . .	256

vtss_restart_status_t	Restart status . . . . .	258
vtss_routing_entry_t	Routing entry . . . . .	259
vtss_secure_on_passwd_t	Structure for Wake-On-LAN Secure-On Password . . . . .	260
vtss_serdes_macro_conf_t	Serdes macro configuration . . . . .	261
vtss_sflow_port_conf_t	SFlow configuration structure . . . . .	262
vtss_sgpi_conf_t	SGPIO configuration for a group . . . . .	263
vtss_sgpi_port_conf_t	SGPIO port configuration . . . . .	265
vtss_sgpi_port_data_t	SGPIO read data for a port . . . . .	266
vtss_shaper_t	Shaper . . . . .	267
vtss_sync_clock_in_t	Struct containing configuration selecting the recovered input clock port, to be delivered to a selected output clock port . . . . .	268
vtss_sync_clock_out_t	Struct containing configuration for a recovered clock output port . . . . .	269
vtss_tci_t	Tag Control Information (according to IEEE 802.1Q) . . . . .	270
vtss_timeofday_t	Time of day structure . . . . .	271
vtss_timestamp_t	Time stamp in seconds and nanoseconds . . . . .	272
vtss_trace_conf_t	Trace group configuration . . . . .	273
vtss_ts_ext_clock_mode_t	External clock output configuration . . . . .	274
vtss_ts_id_t	Timestamp identifier . . . . .	275
vtss_ts_internal_mode_t	Hardware timestamping format mode for internal ports . . . . .	276
vtss_ts_operation_mode_t	Timestamp operation . . . . .	277
vtss_ts_timestamp_alloc_t	Timestamp allocation . . . . .	277
vtss_ts_timestamp_t	Timestamp structure . . . . .	278
vtss_vcap_ip_t	VCAP IPv4 address value and mask . . . . .	280
vtss_vcap_u128_t	VCAP 128 bit value and mask . . . . .	281
vtss_vcap_u16_t	VCAP 16 bit value and mask . . . . .	282
vtss_vcap_u24_t	VCAP 24 bit value and mask . . . . .	283
vtss_vcap_u32_t	VCAP 32 bit value and mask . . . . .	284
vtss_vcap_u40_t	VCAP 40 bit value and mask . . . . .	285
vtss_vcap_u48_t	VCAP 48 bit value and mask . . . . .	286

<a href="#">vtss_vcap_u8_t</a>	VCAP 8 bit value and mask . . . . .	287
<a href="#">vtss_vcap_udp_tcp_t</a>	VCAP UDP/TCP port range . . . . .	288
<a href="#">vtss_vcap_vid_t</a>	VCAP VLAN ID value and mask . . . . .	289
<a href="#">vtss_vcap_vr_t</a>	VCAP universal value or range . . . . .	290
<a href="#">vtss_vce_action_t</a>	VCE Action . . . . .	292
<a href="#">vtss_vce_frame_etype_t</a>	Frame data for VTSS_VCE_TYPE_ETYPE . . . . .	293
<a href="#">vtss_vce_frame_ipv4_t</a>	Frame data for VTSS_VCE_TYPE_IPV4 . . . . .	294
<a href="#">vtss_vce_frame_ipv6_t</a>	Frame data for VTSS_VCE_TYPE_IPV6 . . . . .	296
<a href="#">vtss_vce_frame_llc_t</a>	Frame data for VTSS_VCE_TYPE_LLCC . . . . .	297
<a href="#">vtss_vce_frame_snap_t</a>	Frame data for VTSS_VCE_TYPE_SNAP . . . . .	298
<a href="#">vtss_vce_key_t</a>	VCE Key . . . . .	299
<a href="#">vtss_vce_mac_t</a>	VCE MAC header information . . . . .	301
<a href="#">vtss_vce_t</a>	VLAN Control Entry . . . . .	302
<a href="#">vtss_vce_tag_t</a>	VCE tag information . . . . .	303
<a href="#">vtss_vcl_port_conf_t</a>	VCL port configuration . . . . .	305
<a href="#">vtss_vid_mac_t</a>	MAC Address in specific VLAN . . . . .	306
<a href="#">vtss_vlan_conf_t</a>	VLAN configuration . . . . .	307
<a href="#">vtss_vlan_port_conf_t</a>	VLAN port configuration . . . . .	308
<a href="#">vtss_vlan_tag_t</a>	. . . . .	309
<a href="#">vtss_vlan_trans_grp2vlan_conf_t</a>	VLAN translation group-to-VLAN configuration . . . . .	311
<a href="#">vtss_vlan_trans_port2grp_conf_t</a>	VLAN translation port-to-group configuration . . . . .	312
<a href="#">vtss_vlan_vid_conf_t</a>	VLAN ID configuration . . . . .	313
<a href="#">vtss_wol_mac_addr_t</a>	Structure for Wake-On-LAN MAC Address . . . . .	314



# Chapter 5

## File Index

### 5.1 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 . . . . .	384
vtss_api/include/ <a href="#">vtss_afi_api.h</a>	
AFI API . . . . .	384
vtss_api/include/ <a href="#">vtss_aneg_api.h</a>	
ANEG API . . . . .	384
vtss_api/include/ <a href="#">vtss_api.h</a>	
Vitesse API main header file . . . . .	385
vtss_api/include/ <a href="#">vtss_evc_api.h</a>	
EVC API . . . . .	385
vtss_api/include/ <a href="#">vtss_fdma_api.h</a>	
Frame DMA API . . . . .	395
vtss_api/include/ <a href="#">vtss_gfp_api.h</a>	
GFP API . . . . .	396
vtss_api/include/ <a href="#">vtss_hqos_api.h</a>	
HQoS API . . . . .	396
vtss_api/include/ <a href="#">vtss_i2c_api.h</a>	
I2C API . . . . .	396
vtss_api/include/ <a href="#">vtss_init_api.h</a>	
Initialization API . . . . .	397
vtss_api/include/ <a href="#">vtss_l2_api.h</a>	
Layer 2 API . . . . .	410
vtss_api/include/ <a href="#">vtss_l3_api.h</a>	
L3 routing API . . . . .	468
vtss_api/include/ <a href="#">vtss_mac10g_api.h</a>	
MAC10G API . . . . .	468
vtss_api/include/ <a href="#">vtss_macsec_api.h</a>	
MACsec API . . . . .	??
vtss_api/include/ <a href="#">vtss_misc_api.h</a>	
Miscellaneous API . . . . .	468
vtss_api/include/ <a href="#">vtss_mpls_api.h</a>	
MPLS API . . . . .	503
vtss_api/include/ <a href="#">vtss_oam_api.h</a>	
OAM API . . . . .	503
vtss_api/include/ <a href="#">vtss_oha_api.h</a>	
OHA API . . . . .	503

vtss_api/include/vtss_os.h	
OS Layer API . . . . .	504
vtss_api/include/vtss_os_custom.h	
OS custom header file . . . . .	504
vtss_api/include/vtss_os_ecos.h	
ECos OS API . . . . .	509
vtss_api/include/vtss_os_linux.h	
Linux OS API . . . . .	519
vtss_api/include/vtss_otn_api.h	
OTN API . . . . .	526
vtss_api/include/vtss_packet_api.h	
Packet API . . . . .	526
vtss_api/include/vtss_pcs_10gbase_r_api.h	
PCS_10BASE_R API . . . . .	548
vtss_api/include/vtss_phy_10g_api.h	
10G PHY API . . . . .	548
vtss_api/include/vtss_phy_api.h	
PHY API . . . . .	548
vtss_api/include/vtss_phy_ts_api.h	
PHY TimeStamping API . . . . .	625
vtss_api/include/vtss_port_api.h	
Port API . . . . .	626
vtss_api/include/vtss_qos_api.h	
QoS API . . . . .	638
vtss_api/include/vtss_rab_api.h	
RAB API . . . . .	647
vtss_api/include/vtss_security_api.h	
Security API . . . . .	647
vtss_api/include/vtss_sfi4_api.h	
SFI4 API . . . . .	657
vtss_api/include/vtss_sync_api.h	
Synchronization API . . . . .	658
vtss_api/include/vtss_tfi5_api.h	
TFI5 API . . . . .	661
vtss_api/include/vtss_ts_api.h	
TimeStamping API . . . . .	662
vtss_api/include/vtss_upi_api.h	
Define UPI API interface . . . . .	678
vtss_api/include/vtss_wis_api.h	
EWIS layer API . . . . .	678
vtss_api/include/vtss_xaui_api.h	
XAUI API . . . . .	678
vtss_api/include/vtss_xfi_api.h	
XFI API . . . . .	679
vtss_api/include/vtss/api/l2_types.h	
Layer 2 Public API Header for l2 . . . . .	315
vtss_api/include/vtss/api/options.h	
Features and options . . . . .	316
vtss_api/include/vtss/api/phy.h	
PHY Public API Header . . . . .	332
vtss_api/include/vtss/api/port.h	
Port Public API Header . . . . .	334
vtss_api/include/vtss/api/types.h	
Generic types API . . . . .	347

# Chapter 6

## Data Structure Documentation

### 6.1 port\_custom\_conf\_t Struct Reference

Port configuration.

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

#### Data Fields

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

#### 6.1.1 Detailed Description

Port configuration.

Definition at line 269 of file port.h.

#### 6.1.2 Field Documentation

### 6.1.2.1 enable

`BOOL port_custom_conf_t::enable`

Admin enable/disable

Definition at line 270 of file port.h.

### 6.1.2.2 autoneg

`BOOL port_custom_conf_t::autoneg`

Auto negotiation

Definition at line 271 of file port.h.

### 6.1.2.3 fdx

`BOOL port_custom_conf_t::fdx`

Forced duplex mode

Definition at line 272 of file port.h.

### 6.1.2.4 flow\_control

`BOOL port_custom_conf_t::flow_control`

Flow control (Standard 802.3x)

Definition at line 273 of file port.h.

### 6.1.2.5 pfc

`BOOL port_custom_conf_t::pfc[VTSS_PRIOS]`

Priority Flow control (802.1Qbb)

Definition at line 275 of file port.h.

### 6.1.2.6 speed

`vtss_port_speed_t` `port_custom_conf_t::speed`

Forced port speed

Definition at line 277 of file port.h.

### 6.1.2.7 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.

### 6.1.2.8 max\_length

`unsigned int` `port_custom_conf_t::max_length`

Max frame length

Definition at line 279 of file port.h.

### 6.1.2.9 exc\_col\_cont

`BOOL` `port_custom_conf_t::exc_col_cont`

Excessive collision continuation

Definition at line 283 of file port.h.

### 6.1.2.10 adv\_dis

`u8` `port_custom_conf_t::adv_dis`

Auto neg advertisement disable

Definition at line 284 of file port.h.

### 6.1.2.11 max\_tags

`u8 port_custom_conf_t::max_tags`

Maximum number of tags

Definition at line 285 of file port.h.

### 6.1.2.12 oper\_up

`BOOL port_custom_conf_t::oper_up`

Force operational state up

Definition at line 286 of file port.h.

### 6.1.2.13 frame\_length\_chk

`BOOL port_custom_conf_t::frame_length_chk`

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

Definition at line 287 of file port.h.

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

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

## 6.2 serdes\_fields\_t Struct Reference

Serdes fields.

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

### Data Fields

- `u32 ob_post0`
- `u32 ob_sr`

### 6.2.1 Detailed Description

Serdes fields.

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

## 6.2.2 Field Documentation

### 6.2.2.1 ob\_post0

`u32 serdes_fields_t::ob_post0`

Trace length

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

### 6.2.2.2 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`

## 6.3 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`

### 6.3.1 Detailed Description

Frame data for VTSS\_ACE\_TYPE\_ARP.

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

### 6.3.2 Field Documentation

#### 6.3.2.1 smac

`vtss_ace_u48_t vtss_ace_frame_arp_t::smac`

SMAC

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

#### 6.3.2.2 arp

`vtss_ace_bit_t vtss_ace_frame_arp_t::arp`

Opcode ARP/RARP

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

#### 6.3.2.3 req

`vtss_ace_bit_t vtss_ace_frame_arp_t::req`

Opcode request/reply

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

#### 6.3.2.4 unknown

`vtss_ace_bit_t vtss_ace_frame_arp_t::unknown`

Opcode unknown

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

### 6.3.2.5 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.

### 6.3.2.6 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.

### 6.3.2.7 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.

### 6.3.2.8 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.

### 6.3.2.9 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.

### 6.3.2.10 sip

`vtss_ace_ip_t vtss_ace_frame_arp_t::sip`

Sender IP address

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

### 6.3.2.11 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`

## 6.4 `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`

### 6.4.1 Detailed Description

Frame data for `VTSS_ACE_TYPE_ETYPE`.

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

### 6.4.2 Field Documentation

#### 6.4.2.1 dmac

`vtss_ace_u48_t vtss_ace_frame_etype_t::dmac`

DMAC

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

#### 6.4.2.2 smac

`vtss_ace_u48_t vtss_ace_frame_etype_t::smac`

SMAC

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

#### 6.4.2.3 etype

`vtss_ace_u16_t vtss_ace_frame_etype_t::etype`

Ethernet Type value

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

#### 6.4.2.4 data

`vtss_ace_u16_t vtss_ace_frame_etype_t::data`

MAC data

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

#### 6.4.2.5 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](#)

## 6.5 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](#)

### 6.5.1 Detailed Description

Frame data for VTSS\_ACE\_TYPE\_IPV4.

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

### 6.5.2 Field Documentation

#### 6.5.2.1 ttl

```
vtss_ace_bit_t vtss_ace_frame_ipv4_t::ttl
```

TTL zero

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

### 6.5.2.2 fragment

`vtss_ace_bit_t` `vtss_ace_frame_ipv4_t::fragment`

Fragment

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

### 6.5.2.3 options

`vtss_ace_bit_t` `vtss_ace_frame_ipv4_t::options`

Header options

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

### 6.5.2.4 ds

`vtss_ace_u8_t` `vtss_ace_frame_ipv4_t::ds`

DS field

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

### 6.5.2.5 proto

`vtss_ace_u8_t` `vtss_ace_frame_ipv4_t::proto`

Protocol

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

### 6.5.2.6 sip

`vtss_ace_ip_t` `vtss_ace_frame_ipv4_t::sip`

Source IP address

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

### 6.5.2.7 dip

`vtss_ace_ip_t` `vtss_ace_frame_ipv4_t::dip`

Destination IP address

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

### 6.5.2.8 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.

### 6.5.2.9 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.

### 6.5.2.10 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.

### 6.5.2.11 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.

### 6.5.2.12 `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`.

### 6.5.2.13 `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`.

### 6.5.2.14 `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`.

### 6.5.2.15 `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`.

### 6.5.2.16 `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`.

### 6.5.2.17 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`.

### 6.5.2.18 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`.

### 6.5.2.19 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`.

### 6.5.2.20 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`.

### 6.5.2.21 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`

## 6.6 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](#)

### 6.6.1 Detailed Description

Frame data for VTSS\_ACE\_TYPE\_IPV6.

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

### 6.6.2 Field Documentation

#### 6.6.2.1 proto

```
vtss_ace_u8_t vtss_ace_frame_ipv6_t::proto
```

IPv6 protocol

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

### 6.6.2.2 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.

### 6.6.2.3 ttl

`vtss_ace_bit_t vtss_ace_frame_ipv6_t::ttl`

TTL zero

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

### 6.6.2.4 ds

`vtss_ace_u8_t vtss_ace_frame_ipv6_t::ds`

DS field

Definition at line 499 of file vtss\_security\_api.h.

### 6.6.2.5 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.

### 6.6.2.6 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.

### 6.6.2.7 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`.

### 6.6.2.8 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`.

### 6.6.2.9 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`.

### 6.6.2.10 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`.

### 6.6.2.11 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`.

### 6.6.2.12 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`.

### 6.6.2.13 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`.

### 6.6.2.14 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`.

### 6.6.2.15 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`.

### 6.6.2.16 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`.

### 6.6.2.17 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`

## 6.7 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`

### 6.7.1 Detailed Description

Frame data for `VTSS_ACE_TYPE_LLCC`.

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

### 6.7.2 Field Documentation

#### 6.7.2.1 dmac

`vtss_ace_u48_t` `vtss_ace_frame_llc_t::dmac`

DMAC

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

### 6.7.2.2 smac

`vtss_ace_u48_t vtss_ace_frame_llc_t::smac`

SMAC

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

### 6.7.2.3 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`

## 6.8 `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`

### 6.8.1 Detailed Description

Frame data for VTSS\_ACE\_TYPE\_SNAP.

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

### 6.8.2 Field Documentation

### 6.8.2.1 dmac

`vtss_ace_u48_t vtss_ace_frame_snap_t::dmac`

DMAC

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

### 6.8.2.2 smac

`vtss_ace_u48_t vtss_ace_frame_snap_t::smac`

SMAC

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

### 6.8.2.3 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](#)

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

PTP header filtering.

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

### Data Fields

- `BOOL enable`
- `vtss_ace_u32_t header`

### 6.9.1 Detailed Description

PTP header filtering.

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

## 6.9.2 Field Documentation

### 6.9.2.1 enable

`BOOL vtss_ace_ptp_t::enable`

Enable PTP header filtering

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

### 6.9.2.2 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`

## 6.10 `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`

### 6.10.1 Detailed Description

SIP/SMAC filtering.

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

## 6.10.2 Field Documentation

### 6.10.2.1 enable

`BOOL vtss_ace_sip_smac_t::enable`

Enable SIP/SMAC filtering

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

### 6.10.2.2 sip

`vtss_ip_t vtss_ace_sip_smac_t::sip`

SIP

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

### 6.10.2.3 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`

## 6.11 `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`

### 6.11.1 Detailed Description

Access Control Entry.

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

### 6.11.2 Field Documentation

#### 6.11.2.1 id

`vtss_ace_id_t vtss_ace_t::id`

ACE ID, must be different from VTSS\_ACE\_ID\_LAST

Definition at line 520 of file vtss\_security\_api.h.

#### 6.11.2.2 port\_list

`BOOL vtss_ace_t::port_list [VTSS_PORT_ARRAY_SIZE]`

Port list

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

### 6.11.2.3 policy

`vtss_ace_u8_t vtss_ace_t::policy`

Policy number

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

### 6.11.2.4 type

`vtss_ace_type_t vtss_ace_t::type`

ACE frame type

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

### 6.11.2.5 action

`vtss_acl_action_t vtss_ace_t::action`

ACE action

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

### 6.11.2.6 dmac\_mc

`vtss_ace_bit_t vtss_ace_t::dmac_mc`

Multicast DMAC

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

### 6.11.2.7 dmac\_bc

`vtss_ace_bit_t vtss_ace_t::dmac_bc`

Broadcast DMAC

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

### 6.11.2.8 vlan

`vtss_ace_vlan_t` `vtss_ace_t::vlan`

VLAN Tag

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

### 6.11.2.9 etype

`vtss_ace_frame_etype_t` `vtss_ace_t::etype`

VTSS\_ACE\_TYPE\_ETYPE

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

### 6.11.2.10 llc

`vtss_ace_frame_llc_t` `vtss_ace_t::llc`

VTSS\_ACE\_TYPE\_LLCC

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

### 6.11.2.11 snap

`vtss_ace_frame_snap_t` `vtss_ace_t::snap`

VTSS\_ACE\_TYPE\_SNAP

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

### 6.11.2.12 arp

`vtss_ace_frame_arp_t` `vtss_ace_t::arp`

VTSS\_ACE\_TYPE\_ARP

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

### 6.11.2.13 ipv4

```
vtss_ace_frame_ipv4_t vtss_ace_t::ipv4
```

VTSS\_ACE\_TYPE\_IPV4

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

### 6.11.2.14 ipv6

```
vtss_ace_frame_ipv6_t vtss_ace_t::ipv6
```

VTSS\_ACE\_TYPE\_IPV6

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

### 6.11.2.15 frame

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

Frame type specific data

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

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

## 6.12 vtss\_ace\_vlan\_t Struct Reference

ACE VLAN information.

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

### Data Fields

- [vtss\\_ace\\_vid\\_t vid](#)
- [vtss\\_ace\\_u8\\_t usr\\_prio](#)
- [vtss\\_ace\\_bit\\_t cfi](#)
- [vtss\\_ace\\_bit\\_t tagged](#)

### 6.12.1 Detailed Description

ACE VLAN information.

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

## 6.12.2 Field Documentation

### 6.12.2.1 vid

`vtss_ace_vid_t` `vtss_ace_vlan_t::vid`

VLAN ID (12 bit)

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

### 6.12.2.2 usr\_prio

`vtss_ace_u8_t` `vtss_ace_vlan_t::usr_prio`

User priority/PCP (3 bit)

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

### 6.12.2.3 cfi

`vtss_ace_bit_t` `vtss_ace_vlan_t::cfi`

CFI/DEI

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

### 6.12.2.4 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`

## 6.13 vtss\_acl\_action\_t Struct Reference

ACL Action.

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

### Data Fields

- `BOOL cpu`
- `BOOL cpu_once`
- `vtss_packet_rx_queue_t cpu_queue`
- `BOOL police`
- `vtss_acl_policer_no_t policer_no`
- `BOOL evc_police`
- `vtss_evc_policer_id_t evc_policer_id`
- `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`

#### 6.13.1 Detailed Description

ACL Action.

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

#### 6.13.2 Field Documentation

##### 6.13.2.1 `cpu`

```
BOOL vtss_acl_action_t::cpu
```

Forward to CPU

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

##### 6.13.2.2 `cpu_once`

```
BOOL vtss_acl_action_t::cpu_once
```

Only first frame forwarded to CPU

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

### 6.13.2.3 cpu\_queue

`vtss_packet_rx_queue_t` `vtss_acl_action_t::cpu_queue`

CPU queue

Definition at line 242 of file vtss\_security\_api.h.

### 6.13.2.4 police

`BOOL` `vtss_acl_action_t::police`

Enable policer

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

### 6.13.2.5 policer\_no

`vtss_acl_policer_no_t` `vtss_acl_action_t::policer_no`

Policer number

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

### 6.13.2.6 evc\_police

`BOOL` `vtss_acl_action_t::evc_police`

Enable EVC policer

Definition at line 247 of file vtss\_security\_api.h.

### 6.13.2.7 evc\_policer\_id

`vtss_evc_policer_id_t` `vtss_acl_action_t::evc_policer_id`

EVC policer ID

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

### 6.13.2.8 learn

`BOOL vtss_acl_action_t::learn`

Allow learning

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

### 6.13.2.9 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.

### 6.13.2.10 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.

### 6.13.2.11 mirror

`BOOL vtss_acl_action_t::mirror`

Enable mirroring

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

### 6.13.2.12 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.

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

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

## 6.14 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`

#### 6.14.1 Detailed Description

ACL policer configuration.

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

#### 6.14.2 Field Documentation

##### 6.14.2.1 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`.

##### 6.14.2.2 bit\_rate

```
vtss_bitrate_t vtss_acl_policer_conf_t::bit_rate
```

Bit rate

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

### 6.14.2.3 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](#)

## 6.15 vtss\_acl\_port\_conf\_t Struct Reference

ACL port configuration.

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

### Data Fields

- [vtss\\_acl\\_policy\\_no\\_t](#) policy\_no
- [vtss\\_acl\\_action\\_t](#) action

#### 6.15.1 Detailed Description

ACL port configuration.

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

#### 6.15.2 Field Documentation

##### 6.15.2.1 policy\_no

```
vtss_acl_policy_no_t vtss_acl_port_conf_t::policy_no
```

Policy number

Definition at line 278 of file vtss\_security\_api.h.

### 6.15.2.2 action

```
vtss_acl_action_t vtss_acl_port_conf_t::action
```

Action

Definition at line 279 of file vtss\_security\_api.h.

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

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

## 6.16 vtss\_aggr\_mode\_t Struct Reference

Aggregation traffic distribution mode.

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

### Data Fields

- [BOOL smac\\_enable](#)
- [BOOL dmac\\_enable](#)
- [BOOL sip\\_dip\\_enable](#)
- [BOOL sport\\_dport\\_enable](#)

### 6.16.1 Detailed Description

Aggregation traffic distribution mode.

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

### 6.16.2 Field Documentation

#### 6.16.2.1 smac\_enable

```
BOOL vtss_aggr_mode_t::smac_enable
```

Source MAC address

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

### 6.16.2.2 dmac\_enable

`BOOL vtss_aggr_mode_t::dmac_enable`

Destination MAC address

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

### 6.16.2.3 sip\_dip\_enable

`BOOL vtss_aggr_mode_t::sip_dip_enable`

Source and destination IP address

Definition at line 43 of file l2\_types.h.

### 6.16.2.4 sport\_dport\_enable

`BOOL vtss_aggr_mode_t::sport_dport_enable`

Source and destination UDP/TCP port

Definition at line 44 of file l2\_types.h.

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

- [vtss\\_api/include/vtss/api/l2\\_types.h](#)

## 6.17 vtss\_aneg\_t Struct Reference

Auto negotiation struct.

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

### Data Fields

- [BOOL obey\\_pause](#)
- [BOOL generate\\_pause](#)

### 6.17.1 Detailed Description

Auto negotiation struct.

Definition at line 482 of file types.h.

---

### 6.17.2 Field Documentation

#### 6.17.2.1 obey\_pause

`BOOL vtss_aneg_t::obey_pause`

This port should obey PAUSE frames

Definition at line 484 of file types.h.

#### 6.17.2.2 generate\_pause

`BOOL vtss_aneg_t::generate_pause`

Link partner obeys PAUSE frames

Definition at line 485 of file types.h.

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

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

## 6.18 vtss\_api\_lock\_t Struct Reference

API lock structure.

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

### Data Fields

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

#### 6.18.1 Detailed Description

API lock structure.

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

## 6.18.2 Field Documentation

### 6.18.2.1 inst

`vtss_inst_t vtss_api_lock_t::inst`

Target instance reference

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

### 6.18.2.2 function

`const char* vtss_api_lock_t::function`

Function name

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

### 6.18.2.3 file

`const char* vtss_api_lock_t::file`

File name

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

### 6.18.2.4 line

`int vtss_api_lock_t::line`

Line number

Definition at line 289 of file vtss\_misc\_api.h.

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

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

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

Basic counters structure.

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

### Data Fields

- `u32 rx_frames`
- `u32 tx_frames`

#### 6.19.1 Detailed Description

Basic counters structure.

Definition at line 377 of file `vtss_port_api.h`.

#### 6.19.2 Field Documentation

##### 6.19.2.1 rx\_frames

```
u32 vtss_basic_counters_t::rx_frames
```

Rx frames

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

##### 6.19.2.2 tx\_frames

```
u32 vtss_basic_counters_t::tx_frames
```

Tx frames

Definition at line 380 of file `vtss_port_api.h`.

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

- `vtss_api/include/vtss_port_api.h`

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

Chip ID.

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

## Data Fields

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

### 6.20.1 Detailed Description

Chip ID.

Definition at line 401 of file [vtss\\_misc\\_api.h](#).

### 6.20.2 Field Documentation

#### 6.20.2.1 part\_number

[u16 vtss\\_chip\\_id\\_t::part\\_number](#)

BCD encoded part number

Definition at line 403 of file [vtss\\_misc\\_api.h](#).

#### 6.20.2.2 revision

[u16 vtss\\_chip\\_id\\_t::revision](#)

Chip revision

Definition at line 404 of file [vtss\\_misc\\_api.h](#).

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

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

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

Counter pair.

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

## Data Fields

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

### 6.21.1 Detailed Description

Counter pair.

Definition at line 1111 of file types.h.

### 6.21.2 Field Documentation

#### 6.21.2.1 frames

`vtss_counter_t vtss_counter_pair_t::frames`

Number of frames

Definition at line 1112 of file types.h.

#### 6.21.2.2 bytes

`vtss_counter_t vtss_counter_pair_t::bytes`

Number of bytes

Definition at line 1113 of file types.h.

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

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

## 6.22 vtss\_debug\_info\_t Struct Reference

Debug information structure.

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

### Data Fields

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

### 6.22.1 Detailed Description

Debug information structure.

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

### 6.22.2 Field Documentation

#### 6.22.2.1 layer

`vtss_debug_layer_t` vtss\_debug\_info\_t::layer

Layer

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

#### 6.22.2.2 group

`vtss_debug_group_t` vtss\_debug\_info\_t::group

Function group

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

#### 6.22.2.3 chip\_no

`vtss_chip_no_t` vtss\_debug\_info\_t::chip\_no

Chip number, multi-chip targets

Definition at line 246 of file vtss\_misc\_api.h.

#### 6.22.2.4 port\_list

`BOOL` vtss\_debug\_info\_t::port\_list[`VTSS_PORT_ARRAY_SIZE`]

Port list

Definition at line 247 of file vtss\_misc\_api.h.

### 6.22.2.5 full

`BOOL vtss_debug_info_t::full`

Full information dump

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

### 6.22.2.6 clear

`BOOL vtss_debug_info_t::clear`

Clear counters

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

### 6.22.2.7 vml\_format

`BOOL vtss_debug_info_t::vml_format`

VML format register dump

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

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

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

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

API debug lock structure.

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

### Data Fields

- [vtss\\_chip\\_no\\_t chip\\_no](#)

### 6.23.1 Detailed Description

API debug lock structure.

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

### 6.23.2 Field Documentation

#### 6.23.2.1 chip\_no

`vtss_chip_no_t vtss_debug_lock_t::chip_no`

Chip number (if multi-chip instance).

Definition at line 308 of file vtss\_misc\_api.h.

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

- `vtss_api/include/vtss_misc_api.h`

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

Destination group port configuration.

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

### Data Fields

- `vtss_dgroup_no_t dgroup_no`

#### 6.24.1 Detailed Description

Destination group port configuration.

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

### 6.24.2 Field Documentation

#### 6.24.2.1 dgroup\_no

`vtss_dgroup_no_t vtss_dgroup_port_conf_t::dgroup_no`

Destination port group

Definition at line 1395 of file vtss\_l2\_api.h.

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

- `vtss_api/include/vtss_l2_api.h`

## 6.25 vtss\_dlb\_policer\_conf\_t Struct Reference

Dual leaky buckets policer configuration.

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

### Data Fields

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

### 6.25.1 Detailed Description

Dual leaky buckets policer configuration.

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

### 6.25.2 Field Documentation

#### 6.25.2.1 type

```
vtss_policer_type_t vtss_dlb_policer_conf_t::type
```

Policer type

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

#### 6.25.2.2 enable

```
BOOL vtss_dlb_policer_conf_t::enable
```

Enable/disable policer

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

### 6.25.2.3 cf

`BOOL vtss_dlb_policer_conf_t::cf`

Coupling Flag

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

### 6.25.2.4 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.

### 6.25.2.5 cir

`vtss_bitrate_t vtss_dlb_policer_conf_t::cir`

Committed Information Rate

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

### 6.25.2.6 cbs

`vtss_burst_level_t vtss_dlb_policer_conf_t::cbs`

Committed Burst Size

Definition at line 246 of file vtss\_qos\_api.h.

### 6.25.2.7 eir

`vtss_bitrate_t vtss_dlb_policer_conf_t::eir`

Excess Information Rate

Definition at line 247 of file vtss\_qos\_api.h.

### 6.25.2.8 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`

## 6.26 `vtss_ece_action_t` Struct Reference

ECE action.

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

### Data Fields

- `vtss_ece_dir_t` `dir`
- `vtss_ece_pop_tag_t` `pop_tag`
- `vtss_ece_outer_tag_t` `outer_tag`
- `vtss_evc_id_t` `evc_id`
- `vtss_acl_policy_no_t` `policy_no`
- `BOOL` `prio_enable`
- `vtss_prio_t` `prio`

### 6.26.1 Detailed Description

ECE action.

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

### 6.26.2 Field Documentation

#### 6.26.2.1 dir

`vtss_ece_dir_t` `vtss_ece_action_t::dir`

Traffic direction

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

### 6.26.2.2 pop\_tag

`vtss_ece_pop_tag_t` vtss\_ece\_action\_t::pop\_tag

Ingress VLAN popping

Definition at line 563 of file vtss\_evc\_api.h.

### 6.26.2.3 outer\_tag

`vtss_ece_outer_tag_t` vtss\_ece\_action\_t::outer\_tag

Egress outer VLAN tag (always present)

Definition at line 564 of file vtss\_evc\_api.h.

### 6.26.2.4 evc\_id

`vtss_evc_id_t` vtss\_ece\_action\_t::evc\_id

EVC ID

Definition at line 569 of file vtss\_evc\_api.h.

### 6.26.2.5 policy\_no

`vtss_acl_policy_no_t` vtss\_ece\_action\_t::policy\_no

ACL policy number

Definition at line 570 of file vtss\_evc\_api.h.

### 6.26.2.6 prio\_enable

`BOOL` vtss\_ece\_action\_t::prio\_enable

Enable priority classification

Definition at line 572 of file vtss\_evc\_api.h.

### 6.26.2.7 prio

`vtss_prio_t vtss_ece_action_t::prio`

Priority (QoS class)

Definition at line 573 of file `vtss_evc_api.h`.

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

- `vtss_api/include/vtss_evc_api.h`

## 6.27 `vtss_ece_frame_ipv4_t` Struct Reference

ECE IPv4 information.

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

### Data Fields

- `vtss_vcap_vr_t dscp`
- `vtss_vcap_bit_t fragment`
- `vtss_vcap_u8_t proto`
- `vtss_vcap_ip_t sip`
- `vtss_vcap_vr_t sport`
- `vtss_vcap_vr_t dport`

### 6.27.1 Detailed Description

ECE IPv4 information.

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

### 6.27.2 Field Documentation

#### 6.27.2.1 dscp

`vtss_vcap_vr_t vtss_ece_frame_ipv4_t::dscp`

DSCP field (6 bit)

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

### 6.27.2.2 fragment

`vtss_vcap_bit_t` `vtss_ece_frame_ipv4_t::fragment`

Fragment

Definition at line 464 of file vtss\_evc\_api.h.

### 6.27.2.3 proto

`vtss_vcap_u8_t` `vtss_ece_frame_ipv4_t::proto`

Protocol

Definition at line 465 of file vtss\_evc\_api.h.

### 6.27.2.4 sip

`vtss_vcap_ip_t` `vtss_ece_frame_ipv4_t::sip`

Source IP address

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

### 6.27.2.5 sport

`vtss_vcap_vr_t` `vtss_ece_frame_ipv4_t::sport`

UDP/TCP: Source port

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

### 6.27.2.6 dport

`vtss_vcap_vr_t` `vtss_ece_frame_ipv4_t::dport`

UDP/TCP: Destination port

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

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

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

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

ECE IPv6 information.

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

### Data Fields

- `vtss_vcap_vr_t dscp`
- `vtss_vcap_u8_t proto`
- `vtss_vcap_u128_t sip`
- `vtss_vcap_vr_t sport`
- `vtss_vcap_vr_t dport`

#### 6.28.1 Detailed Description

ECE IPv6 information.

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

#### 6.28.2 Field Documentation

##### 6.28.2.1 dscp

```
vtss_vcap_vr_t vtss_ece_frame_ipv6_t::dscp
```

DSCP field (6 bit)

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

##### 6.28.2.2 proto

```
vtss_vcap_u8_t vtss_ece_frame_ipv6_t::proto
```

Protocol

Definition at line 479 of file `vtss_evc_api.h`.

#### 6.28.2.3 sip

`vtss_vcap_u128_t vtss_ece_frame_ipv6_t::sip`

Source IP address (32 LSB)

Definition at line 480 of file vtss\_evc\_api.h.

#### 6.28.2.4 sport

`vtss_vcap_vr_t vtss_ece_frame_ipv6_t::sport`

UDP/TCP: Source port

Definition at line 484 of file vtss\_evc\_api.h.

#### 6.28.2.5 dport

`vtss_vcap_vr_t vtss_ece_frame_ipv6_t::dport`

UDP/TCP: Destination port

Definition at line 485 of file vtss\_evc\_api.h.

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

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

## 6.29 vtss\_ece\_key\_t Struct Reference

ECE key.

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

### Data Fields

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

### 6.29.1 Detailed Description

ECE key.

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

### 6.29.2 Field Documentation

#### 6.29.2.1 port\_list

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

UNI port list

Definition at line 492 of file vtss\_evc\_api.h.

#### 6.29.2.2 mac

```
vtss_ece_mac_t vtss_ece_key_t::mac
```

MAC header

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

#### 6.29.2.3 tag

```
vtss_ece_tag_t vtss_ece_key_t::tag
```

Tag

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

#### 6.29.2.4 type

```
vtss_ece_type_t vtss_ece_key_t::type
```

Frame type

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

### 6.29.2.5 ipv4

```
vtss_ece_frame_ipv4_t vtss_ece_key_t::ipv4
```

VTSS\_ECE\_TYPE\_IPV4

Definition at line 511 of file vtss\_evc\_api.h.

### 6.29.2.6 ipv6

```
vtss_ece_frame_ipv6_t vtss_ece_key_t::ipv6
```

VTSS\_ECE\_TYPE\_IPV6

Definition at line 512 of file vtss\_evc\_api.h.

### 6.29.2.7 frame

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

Frame type specific data

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

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

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

ECE MAC information.

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

### Data Fields

- [vtss\\_vcap\\_bit\\_t](#) dmac\_mc
- [vtss\\_vcap\\_bit\\_t](#) dmac\_bc
- [vtss\\_vcap\\_u48\\_t](#) smac

### 6.30.1 Detailed Description

ECE MAC information.

Definition at line 420 of file vtss\_evc\_api.h.

### 6.30.2 Field Documentation

#### 6.30.2.1 dmac\_mc

`vtss_vcap_bit_t` `vtss_ece_mac_t::dmac_mc`

Multicast DMAC

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

#### 6.30.2.2 dmac\_bc

`vtss_vcap_bit_t` `vtss_ece_mac_t::dmac_bc`

Broadcast DMAC

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

#### 6.30.2.3 smac

`vtss_vcap_u48_t` `vtss_ece_mac_t::smac`

SMAC

Definition at line 428 of file `vtss_evc_api.h`.

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

- `vtss_api/include/vtss_evc_api.h`

## 6.31 `vtss_ece_outer_tag_t` Struct Reference

ECE outer tag.

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

### Data Fields

- `BOOL enable`
- `BOOL pcp_dei_preserve`
- `vtss_tagprio_t pcp`
- `vtss_dei_t dei`

### 6.31.1 Detailed Description

ECE outer tag.

Definition at line 517 of file vtss\_evc\_api.h.

### 6.31.2 Field Documentation

#### 6.31.2.1 enable

```
BOOL vtss_ece_outer_tag_t::enable
```

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

Definition at line 519 of file vtss\_evc\_api.h.

#### 6.31.2.2 pcp\_dei\_preserve

```
BOOL vtss_ece_outer_tag_t::pcp_dei_preserve
```

Preserved or explicit PCP/DEI values

Definition at line 527 of file vtss\_evc\_api.h.

#### 6.31.2.3 pcp

```
vtss_tagprio_t vtss_ece_outer_tag_t::pcp
```

PCP value

Definition at line 529 of file vtss\_evc\_api.h.

#### 6.31.2.4 dei

```
vtss_dei_t vtss_ece_outer_tag_t::dei
```

DEI value (ignored if colouring enabled)

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

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

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

## 6.32 vtss\_ece\_t Struct Reference

EVC Control Entry.

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

### Data Fields

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

#### 6.32.1 Detailed Description

EVC Control Entry.

Definition at line 582 of file vtss\_evc\_api.h.

#### 6.32.2 Field Documentation

##### 6.32.2.1 id

```
vtss_ece_id_t vtss_ece_t::id
```

Entry ID

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

##### 6.32.2.2 key

```
vtss_ece_key_t vtss_ece_t::key
```

ECE key

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

### 6.32.2.3 action

```
vtss_ece_action_t vtss_ece_t::action
```

ECE action

Definition at line 585 of file vtss\_evc\_api.h.

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

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

## 6.33 vtss\_ece\_tag\_t Struct Reference

ECE tag information.

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

### Data Fields

- [vtss\\_vcap\\_vr\\_t vid](#)
- [vtss\\_vcap\\_u8\\_t pcp](#)
- [vtss\\_vcap\\_bit\\_t dei](#)
- [vtss\\_vcap\\_bit\\_t tagged](#)
- [vtss\\_vcap\\_bit\\_t s\\_tagged](#)

### 6.33.1 Detailed Description

ECE tag information.

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

### 6.33.2 Field Documentation

#### 6.33.2.1 vid

```
vtss_vcap_vr_t vtss_ece_tag_t::vid
```

VLAN ID (12 bit)

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

### 6.33.2.2 pcp

`vtss_vcap_u8_t vtss_ece_tag_t::pcp`

PCP (3 bit)

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

### 6.33.2.3 dei

`vtss_vcap_bit_t vtss_ece_tag_t::dei`

DEI

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

### 6.33.2.4 tagged

`vtss_vcap_bit_t vtss_ece_tag_t::tagged`

Tagged/untagged frame

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

### 6.33.2.5 s\_tagged

`vtss_vcap_bit_t vtss_ece_tag_t::s_tagged`

S-tagged/C-tagged frame

Definition at line 439 of file `vtss_evc_api.h`.

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

- `vtss_api/include/vtss_evc_api.h`

## 6.34 `vtss_eee_port_conf_t` Struct Reference

EEE port configuration.

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

## Data Fields

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

### 6.34.1 Detailed Description

EEE port configuration.

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

### 6.34.2 Field Documentation

#### 6.34.2.1 eee\_ena

```
BOOL vtss_eee_port_conf_t::eee_ena
```

Enable EEE

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

#### 6.34.2.2 eee\_fast\_queues

```
u8 vtss_eee_port_conf_t::eee_fast_queues
```

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

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

#### 6.34.2.3 tx\_tw

```
u16 vtss_eee_port_conf_t::tx_tw
```

Time from path is activated until frame transmission restarted.

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

#### 6.34.2.4 lp\_advertisement

```
u8 vtss_eee_port_conf_t::lp_advertisement
```

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

Definition at line 1224 of file vtss\_misc\_api.h.

#### 6.34.2.5 optimized\_for\_power

```
BOOL vtss_eee_port_conf_t::optimized_for_power
```

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

Definition at line 1226 of file vtss\_misc\_api.h.

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

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

## 6.35 vtss\_eee\_port\_counter\_t Struct Reference

EEE port counters (JR only)

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

### Data Fields

- [BOOL fill\\_level\\_get](#)
- [u32 fill\\_level\\_thres](#)
- [u32 fill\\_level](#)
- [BOOL tx\\_out\\_bytes\\_get](#)
- [u32 tx\\_out\\_bytes](#)

#### 6.35.1 Detailed Description

EEE port counters (JR only)

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

#### 6.35.2 Field Documentation

### 6.35.2.1 fill\_level\_get

```
BOOL vtss_eee_port_counter_t::fill_level_get
```

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

Definition at line 1248 of file vtss\_misc\_api.h.

### 6.35.2.2 fill\_level\_thres

```
u32 vtss_eee_port_counter_t::fill_level_thres
```

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

Definition at line 1249 of file vtss\_misc\_api.h.

### 6.35.2.3 fill\_level

```
u32 vtss_eee_port_counter_t::fill_level
```

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

Definition at line 1250 of file vtss\_misc\_api.h.

### 6.35.2.4 tx\_out\_bytes\_get

```
BOOL vtss_eee_port_counter_t::tx_out_bytes_get
```

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

Definition at line 1251 of file vtss\_misc\_api.h.

### 6.35.2.5 tx\_out\_bytes

```
u32 vtss_eee_port_counter_t::tx_out_bytes
```

[OUT] Transmitted number of bytes, updated by API if [tx\\_out\\_bytes\\_get](#) is TRUE.

Definition at line 1252 of file vtss\_misc\_api.h.

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

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

## 6.36 vtss\_eee\_port\_state\_t Struct Reference

EEE port state (JR only)

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

### Data Fields

- `vtss_eee_state_select_t select`
- `u32 val`

#### 6.36.1 Detailed Description

EEE port state (JR only)

Definition at line 1239 of file vtss\_misc\_api.h.

#### 6.36.2 Field Documentation

##### 6.36.2.1 select

```
vtss_eee_state_select_t vtss_eee_port_state_t::select
```

State to change.

Definition at line 1241 of file vtss\_misc\_api.h.

##### 6.36.2.2 val

```
u32 vtss_eee_port_state_t::val
```

New value to apply. Interpretation depends on `select`.

Definition at line 1242 of file vtss\_misc\_api.h.

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

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

## 6.37 vtss\_eps\_port\_conf\_t Struct Reference

Port protection configuration.

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

## Data Fields

- `vtss_eps_port_type_t type`
- `vtss_port_no_t port_no`

### 6.37.1 Detailed Description

Port protection configuration.

Definition at line 2141 of file `vtss_l2_api.h`.

### 6.37.2 Field Documentation

#### 6.37.2.1 type

`vtss_eps_port_type_t vtss_eps_port_conf_t::type`

Protection type

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

#### 6.37.2.2 port\_no

`vtss_port_no_t vtss_eps_port_conf_t::port_no`

Protection port or VTSS\_PORT\_NO\_NONE

Definition at line 2144 of file `vtss_l2_api.h`.

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

- `vtss_api/include/vtss_l2_api.h`

## 6.38 vtss\_evc\_conf\_t Struct Reference

EVC configuration (excluding UNIs)

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

## Data Fields

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

### 6.38.1 Detailed Description

EVC configuration (excluding UNIs)

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

### 6.38.2 Field Documentation

#### 6.38.2.1 learning

`BOOL vtss_evc_conf_t::learning`

Enable/disable learning

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

#### 6.38.2.2 pb

`vtss_evc_pb_conf_t vtss_evc_conf_t::pb`

PB specific configuration

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

#### 6.38.2.3 network

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

Network specific configuration

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

- `vtss_api/include/vtss_evc_api.h`

## 6.39 vtss\_evc\_inner\_tag\_t Struct Reference

EVC inner tag.

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

### Data Fields

- `vtss_evc_inner_tag_type_t type`
- `vtss_evc_vid_mode_t vid_mode`
- `vtss_vid_t vid`
- `BOOL pcp_dei_preserve`
- `vtss_tagprio_t pcp`
- `vtss_dei_t dei`

#### 6.39.1 Detailed Description

EVC inner tag.

Definition at line 263 of file vtss\_evc\_api.h.

#### 6.39.2 Field Documentation

##### 6.39.2.1 type

```
vtss_evc_inner_tag_type_t vtss_evc_inner_tag_t::type
```

Tag type

Definition at line 265 of file vtss\_evc\_api.h.

##### 6.39.2.2 vid\_mode

```
vtss_evc_vid_mode_t vtss_evc_inner_tag_t::vid_mode
```

VLAN ID mode

Definition at line 266 of file vtss\_evc\_api.h.

### 6.39.2.3 vid

`vtss_vid_t` `vtss_evc_inner_tag_t::vid`

VLAN ID

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

### 6.39.2.4 pcp\_dei\_preserve

`BOOL` `vtss_evc_inner_tag_t::pcp_dei_preserve`

Preserved or explicit PCP/DEI values

Definition at line 268 of file `vtss_evc_api.h`.

### 6.39.2.5 pcp

`vtss_tagprio_t` `vtss_evc_inner_tag_t::pcp`

PCP value

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

### 6.39.2.6 dei

`vtss_dei_t` `vtss_evc_inner_tag_t::dei`

DEI value

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

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

- `vtss_api/include/vtss_evc_api.h`

## 6.40 vtss\_evc\_pb\_conf\_t Struct Reference

PB specific EVC configuration.

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

## Data Fields

- `BOOL nni [VTSS_PORT_ARRAY_SIZE]`
- `vtss_vid_t ivid`
- `vtss_vid_t vid`
- `vtss_vid_t uvid`
- `vtss_evc_inner_tag_t inner_tag`

### 6.40.1 Detailed Description

PB specific EVC configuration.

Definition at line 275 of file vtss\_evc\_api.h.

### 6.40.2 Field Documentation

#### 6.40.2.1 nni

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

NNI configuration

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

#### 6.40.2.2 ivid

```
vtss_vid_t vtss_evc_pb_conf_t::ivid
```

Internal VID

Definition at line 277 of file vtss\_evc\_api.h.

#### 6.40.2.3 vid

```
vtss_vid_t vtss_evc_pb_conf_t::vid
```

NNI VID of outer tag

Definition at line 278 of file vtss\_evc\_api.h.

#### 6.40.2.4 uvid

`vtss_vid_t vtss_evc_pb_conf_t::uvid`

UNI VID of outer tag (VTSS\_ECE\_DIR\_NNI\_TO\_UNI only)

Definition at line 285 of file `vtss_evc_api.h`.

#### 6.40.2.5 inner\_tag

`vtss_evc_inner_tag_t vtss_evc_pb_conf_t::inner_tag`

Inner tag (optional)

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

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

- `vtss_api/include/vtss_evc_api.h`

## 6.41 `vtss_evc_port_conf_t` Struct Reference

EVC port configuration.

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

### Data Fields

- `BOOL dei_colouring`
- `BOOL inner_tag`
- `BOOL dmac_dip`

#### 6.41.1 Detailed Description

EVC port configuration.

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

#### 6.41.2 Field Documentation

#### 6.41.2.1 dei\_colouring

`BOOL vtss_evc_port_conf_t::dei_colouring`

NNI: Enable colouring of DEI for received frames

Definition at line 152 of file vtss\_evc\_api.h.

#### 6.41.2.2 inner\_tag

`BOOL vtss_evc_port_conf_t::inner_tag`

NNI: Enable inner tag (default outer tag)

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

#### 6.41.2.3 dmac\_dip

`BOOL vtss_evc_port_conf_t::dmac_dip`

UNI: Enable DMAC/DIP matching (default SMAC/SIP)

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

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

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

## 6.42 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

### 6.42.1 Detailed Description

Fan specifications.

Definition at line 1148 of file vtss\_misc\_api.h.

### 6.42.2 Field Documentation

#### 6.42.2.1 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.

#### 6.42.2.2 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.

#### 6.42.2.3 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.

#### 6.42.2.4 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.

#### 6.42.2.5 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](#)

## 6.43 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

### 6.43.1 Detailed Description

Initialization configuration.

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

### 6.43.2 Field Documentation

#### 6.43.2.1 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.

#### 6.43.2.2 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.

#### 6.43.2.3 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.

#### 6.43.2.4 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.

#### 6.43.2.5 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.

#### 6.43.2.6 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.

#### 6.43.2.7 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.

#### 6.43.2.8 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.

#### 6.43.2.9 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.

#### 6.43.2.10 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.

#### 6.43.2.11 warm\_start\_enable

`BOOL vtss_init_conf_t::warm_start_enable`

Allow warm start

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

#### 6.43.2.12 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.

#### 6.43.2.13 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.

#### 6.43.2.14 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.

#### 6.43.2.15 pi

`vtss_pi_conf_t vtss_init_conf_t::pi`

Parallel Interface configuration

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

### 6.43.2.16 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`

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

Create structure.

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

### Data Fields

- `vtss_target_type_t target`

#### 6.44.1 Detailed Description

Create structure.

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

#### 6.44.2 Field Documentation

##### 6.44.2.1 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`

## 6.45 vtss\_intr\_t Struct Reference

Interrupt source structure.

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

### Data Fields

- `BOOL link_change [VTSS_PORT_ARRAY_SIZE]`

#### 6.45.1 Detailed Description

Interrupt source structure.

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

#### 6.45.2 Field Documentation

##### 6.45.2.1 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`

## 6.46 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`

### 6.46.1 Detailed Description

Either an IPv4 or IPv6 address.

Definition at line 813 of file types.h.

### 6.46.2 Field Documentation

#### 6.46.2.1 type

`vtss_ip_type_t vtss_ip_addr_t::type`

Union type

Definition at line 814 of file types.h.

#### 6.46.2.2 ipv4

`vtss_ipv4_t vtss_ip_addr_t::ipv4`

IPv4 address

Definition at line 816 of file types.h.

#### 6.46.2.3 ipv6

`vtss_ipv6_t vtss_ip_addr_t::ipv6`

IPv6 address

Definition at line 817 of file types.h.

#### 6.46.2.4 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`

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

IPv6 network.

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

### Data Fields

- `vtss_ip_addr_t address`
- `vtss_prefix_size_t prefix_size`

#### 6.47.1 Detailed Description

IPv6 network.

Definition at line 836 of file types.h.

#### 6.47.2 Field Documentation

##### 6.47.2.1 address

```
vtss_ip_addr_t vtss_ip_network_t::address
```

Network address

Definition at line 838 of file types.h.

##### 6.47.2.2 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`

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

IPv4 network.

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

## Data Fields

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

### 6.48.1 Detailed Description

IPv4 network.

Definition at line 822 of file types.h.

### 6.48.2 Field Documentation

#### 6.48.2.1 address

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

Network address

Definition at line 824 of file types.h.

#### 6.48.2.2 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](#)

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

IPv4 unicast routing entry.

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

## Data Fields

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

### 6.49.1 Detailed Description

IPv4 unicast routing entry.

Definition at line 852 of file types.h.

### 6.49.2 Field Documentation

#### 6.49.2.1 network

`vtss_ipv4_network_t vtss_ipv4_uc_t::network`

Network to route

Definition at line 854 of file types.h.

#### 6.49.2.2 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`

## 6.50 `vtss_ipv6_network_t` Struct Reference

IPv6 network.

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

### Data Fields

- `vtss_ipv6_t address`
- `vtss_prefix_size_t prefix_size`

### 6.50.1 Detailed Description

IPv6 network.

Definition at line 829 of file types.h.

### 6.50.2 Field Documentation

#### 6.50.2.1 address

`vtss_ipv6_t vtss_ipv6_network_t::address`

Network address

Definition at line 831 of file types.h.

#### 6.50.2.2 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`

## 6.51 vtss\_ipv6\_t Struct Reference

IPv6 address/mask.

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

### Data Fields

- `u8 addr [16]`

### 6.51.1 Detailed Description

IPv6 address/mask.

Definition at line 797 of file types.h.

## 6.51.2 Field Documentation

### 6.51.2.1 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](#)

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

IPv6 routing entry.

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

### Data Fields

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

### 6.52.1 Detailed Description

IPv6 routing entry.

Definition at line 860 of file types.h.

## 6.52.2 Field Documentation

### 6.52.2.1 network

`vtss_ipv6_network_t vtss_ipv6_uc_t::network`

Network to route

Definition at line 862 of file types.h.

### 6.52.2.2 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`

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

Interrupt configuration options.

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

### Data Fields

- `BOOL external`
- `u8 destination`

### 6.53.1 Detailed Description

Interrupt configuration options.

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

### 6.53.2 Field Documentation

#### 6.53.2.1 external

`BOOL vtss_irq_conf_t::external`

Redirect to external IRQ

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

### 6.53.2.2 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`

## 6.54 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`

### 6.54.1 Detailed Description

Interrupt status structure.

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

### 6.54.2 Field Documentation

#### 6.54.2.1 active

`u32 vtss_irq_status_t::active`

Bitmap for pending IRQs (VTSS\_IRQ\_xxx)

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

#### 6.54.2.2 raw\_ident

```
u32 vtss_irq_status_t::raw_ident
```

RAW (target dependentant) bitmap for active pending IRQs

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

#### 6.54.2.3 raw\_status

```
u32 vtss_irq_status_t::raw_status
```

RAW (target dependentant) bitmap for all pending IRQs

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

#### 6.54.2.4 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

## 6.55 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

### 6.55.1 Detailed Description

Routing interface statics counter.

Definition at line 886 of file types.h.

### 6.55.2 Field Documentation

#### 6.55.2.1 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.

#### 6.55.2.2 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.

#### 6.55.2.3 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.

#### 6.55.2.4 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.

### 6.55.2.5 ipv4uc\_transmitted\_octets

`u64 vtss_l3_counters_t::ipv4uc_transmitted_octets`

IPv4UC octets transmitted

Definition at line 892 of file types.h.

### 6.55.2.6 ipv4uc\_transmitted\_frames

`u64 vtss_l3_counters_t::ipv4uc_transmitted_frames`

IPv4UC frames transmitted

Definition at line 893 of file types.h.

### 6.55.2.7 ipv6uc\_transmitted\_octets

`u64 vtss_l3_counters_t::ipv6uc_transmitted_octets`

IPv6UC octets transmitted

Definition at line 894 of file types.h.

### 6.55.2.8 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`

## 6.56 vtss\_lcp\_ll\_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`

### 6.56.1 Detailed Description

Structure for Get PHY LC-PLL status.

Definition at line 1777 of file `vtss_phy_api.h`.

### 6.56.2 Field Documentation

#### 6.56.2.1 lock\_status

`u8 vtss_lcppll_status_t::lock_status`

PLL lock status

Definition at line 1778 of file `vtss_phy_api.h`.

#### 6.56.2.2 cal\_done

`u8 vtss_lcppll_status_t::cal_done`

Calibration status

Definition at line 1779 of file `vtss_phy_api.h`.

#### 6.56.2.3 cal\_error

`u8 vtss_lcppll_status_t::cal_error`

Calibration Error indication

Definition at line 1780 of file `vtss_phy_api.h`.

#### 6.56.2.4 fsm\_lock

`u8 vtss_lcp11_status_t::fsm_lock`

FSM lock status

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

#### 6.56.2.5 fsm\_stat

`u8 vtss_lcp11_status_t::fsm_stat`

FSM internal status

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

#### 6.56.2.6 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](#)

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

Learning mode.

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

### Data Fields

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

#### 6.57.1 Detailed Description

Learning mode.

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

---

## 6.57.2 Field Documentation

### 6.57.2.1 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`.

### 6.57.2.2 cpu

`BOOL vtss_learn_mode_t::cpu`

Learn frames copied to CPU (default disabled)

Definition at line 382 of file `vtss_l2_api.h`.

### 6.57.2.3 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`

## 6.58 vtss\_mac\_t Struct Reference

MAC Address.

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

### Data Fields

- `u8 addr [6]`

### 6.58.1 Detailed Description

MAC Address.

Definition at line 643 of file types.h.

### 6.58.2 Field Documentation

#### 6.58.2.1 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](#)

## 6.59 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](#)

### 6.59.1 Detailed Description

MAC address entry.

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

### 6.59.2 Field Documentation

### 6.59.2.1 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.

### 6.59.2.2 destination

`BOOL vtss_mac_table_entry_t::destination[VTSS_PORT_ARRAY_SIZE]`

Dest. ports

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

### 6.59.2.3 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.

### 6.59.2.4 locked

`BOOL vtss_mac_table_entry_t::locked`

Locked/static flag

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

### 6.59.2.5 aged

`BOOL vtss_mac_table_entry_t::aged`

Age flag

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

### 6.59.2.6 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.

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

- `vtss_api/include/vtss_l2_api.h`

## 6.60 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`

### 6.60.1 Detailed Description

MAC address table status.

Definition at line 358 of file vtss\_l2\_api.h.

### 6.60.2 Field Documentation

#### 6.60.2.1 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.

#### 6.60.2.2 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`.

#### 6.60.2.3 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`.

#### 6.60.2.4 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`

## 6.61 vtss\_mce\_action\_t Struct Reference

MCE action.

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

### Data Fields

- `vtss_acl_policy_no_t policy_no`
- `BOOL prio_enable`
- `vtss_prio_t prio`
- `vtss_vid_t vid`
- `u8 pop_cnt`

### 6.61.1 Detailed Description

MCE action.

Definition at line 759 of file vtss\_evc\_api.h.

### 6.61.2 Field Documentation

#### 6.61.2.1 policy\_no

`vtss_acl_policy_no_t vtss_mce_action_t::policy_no`

ACL policy number

Definition at line 776 of file vtss\_evc\_api.h.

#### 6.61.2.2 prio\_enable

`BOOL vtss_mce_action_t::prio_enable`

Enable priority control

Definition at line 777 of file vtss\_evc\_api.h.

#### 6.61.2.3 prio

`vtss_prio_t vtss_mce_action_t::prio`

Selected priority

Definition at line 778 of file vtss\_evc\_api.h.

#### 6.61.2.4 vid

`vtss_vid_t vtss_mce_action_t::vid`

Replace VID

Definition at line 779 of file vtss\_evc\_api.h.

### 6.61.2.5 pop\_cnt

```
u8 vtss_mce_action_t::pop_cnt
```

Pop count

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

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

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

## 6.62 vtss\_mce\_key\_t Struct Reference

MCE key.

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

### Data Fields

- [BOOL port\\_list \[VTSS\\_PORT\\_ARRAY\\_SIZE\]](#)
- [vtss\\_vcap\\_vid\\_t vid](#)
- [vtss\\_vcap\\_u16\\_t data](#)

### 6.62.1 Detailed Description

MCE key.

Definition at line 682 of file vtss\_evc\_api.h.

### 6.62.2 Field Documentation

#### 6.62.2.1 port\_list

```
BOOL vtss_mce_key_t::port_list [VTSS_PORT_ARRAY_SIZE]
```

Ingress port list

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

### 6.62.2.2 vid

`vtss_vcap_vid_t` `vtss_mce_key_t::vid`

Classified VID

Definition at line 697 of file vtss\_evc\_api.h.

### 6.62.2.3 data

`vtss_vcap_u16_t` `vtss_mce_key_t::data`

Two first data bytes after Ethertype

Definition at line 698 of file vtss\_evc\_api.h.

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

- `vtss_api/include/vtss_evc_api.h`

## 6.63 vtss\_mce\_t Struct Reference

MEP Control Entry.

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

### Data Fields

- `BOOL tt_loop`
- `vtss_mce_id_t id`
- `vtss_mce_key_t key`
- `vtss_mce_action_t action`

### 6.63.1 Detailed Description

MEP Control Entry.

Definition at line 784 of file vtss\_evc\_api.h.

### 6.63.2 Field Documentation

### 6.63.2.1 tt\_loop

`BOOL vtss_mce_t::tt_loop`

This is a TT\_LOOP entry. The TT\_LOOP VCAP user is used when creating entry

Definition at line 786 of file `vtss_evc_api.h`.

### 6.63.2.2 id

`vtss_mce_id_t vtss_mce_t::id`

Entry ID

Definition at line 787 of file `vtss_evc_api.h`.

### 6.63.2.3 key

`vtss_mce_key_t vtss_mce_t::key`

MCE key

Definition at line 788 of file `vtss_evc_api.h`.

### 6.63.2.4 action

`vtss_mce_action_t vtss_mce_t::action`

MCE action

Definition at line 789 of file `vtss_evc_api.h`.

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

- `vtss_api/include/vtss_evc_api.h`

## 6.64 vtss\_mirror\_conf\_t Struct Reference

Mirror configuration.

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

## Data Fields

- `vtss_port_no_t port_no`
- `BOOL fwd_enable`

### 6.64.1 Detailed Description

Mirror configuration.

Definition at line 1681 of file `vtss_l2_api.h`.

### 6.64.2 Field Documentation

#### 6.64.2.1 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`.

#### 6.64.2.2 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`.

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

- `vtss_api/include/vtss_l2_api.h`

## 6.65 vtss\_mtimer\_t Struct Reference

Timer structure.

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

## Data Fields

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

### 6.65.1 Detailed Description

Timer structure.

Definition at line 88 of file vtss\_os\_linux.h.

### 6.65.2 Field Documentation

#### 6.65.2.1 timeout

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

Timeout

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

#### 6.65.2.2 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

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

NPI configuration.

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

### Data Fields

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

### 6.66.1 Detailed Description

NPI configuration.

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

### 6.66.2 Field Documentation

#### 6.66.2.1 enable

`BOOL vtss_npi_conf_t::enable`

Enable NPI port

Definition at line 49 of file vtss\_packet\_api.h.

#### 6.66.2.2 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

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

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

### Data Fields

- unsigned int `hw_cnt`

### 6.67.1 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.

## 6.67.2 Field Documentation

### 6.67.2.1 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](#)

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

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

### Data Fields

- `BOOL dma_enable [VTSS_QUEUE_ARRAY_SIZE]`

### 6.68.1 Detailed Description

The packet DMA configuration.

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

### 6.68.2 Field Documentation

#### 6.68.2.1 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](#)

## 6.69 vtss\_packet\_frame\_info\_t Struct Reference

Information about frame.

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

### Data Fields

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

#### 6.69.1 Detailed Description

Information about frame.

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

#### 6.69.2 Field Documentation

##### 6.69.2.1 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.

##### 6.69.2.2 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.

#### 6.69.2.3 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`.

#### 6.69.2.4 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`.

#### 6.69.2.5 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`

## 6.70 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`

#### 6.70.1 Detailed Description

Packet information for each port.

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

## 6.70.2 Field Documentation

### 6.70.2.1 filter

`vtss_packet_filter_t vtss_packet_port_filter_t::filter`

Packet filtering

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

### 6.70.2.2 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](#)

## 6.71 vtss\_packet\_port\_info\_t Struct Reference

Port info structure.

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

### Data Fields

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

### 6.71.1 Detailed Description

Port info structure.

Definition at line 390 of file vtss\_packet\_api.h.

## 6.71.2 Field Documentation

### 6.71.2.1 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.

### 6.71.2.2 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.

### 6.71.2.3 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.

### 6.71.2.4 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](#)

## 6.72 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]`

### 6.72.1 Detailed Description

CPU Rx configuration.

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

### 6.72.2 Field Documentation

#### 6.72.2.1 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.

#### 6.72.2.2 reg

```
vtss_packet_rx_reg_t vtss_packet_rx_conf_t::reg
```

Packet registration

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

### 6.72.2.3 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`.

### 6.72.2.4 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`

## 6.73 `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`

### 6.73.1 Detailed Description

System frame header describing received frame.

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

### 6.73.2 Field Documentation

### 6.73.2.1 length

`u32 vtss_packet_rx_header_t::length`

Frame length excluding CRC

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

### 6.73.2.2 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.

### 6.73.2.3 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.

### 6.73.2.4 learn

`BOOL vtss_packet_rx_header_t::learn`

TRUE if learn frame

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

### 6.73.2.5 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.

### 6.73.2.6 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`.

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

- `vtss_api/include/vtss_packet_api.h`

## 6.74 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`

### 6.74.1 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`.

## 6.74.2 Field Documentation

### 6.74.2.1 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](#).

### 6.74.2.2 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](#).

### 6.74.2.3 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.

### 6.74.2.4 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.

#### 6.74.2.5 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`.

#### 6.74.2.6 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`.

#### 6.74.2.7 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`.

#### 6.74.2.8 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`.

### 6.74.2.9 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.

### 6.74.2.10 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.

### 6.74.2.11 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.

#### 6.74.2.12 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`.

#### 6.74.2.13 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`.

#### 6.74.2.14 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`.

#### 6.74.2.15 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\_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.

#### 6.74.2.16 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.

#### 6.74.2.17 sflow\_type

`vtss_sflow_type_t vtss_packet_rx_info_t::sflow_type`

sFlow type. Indicates if this is copied to the CPU due to Rx or Tx SFlow, or if it's not due to sFlow in the first place (VTSS\_SFLOW\_TYPE\_NONE).

Only VTSS\_SFLOW\_TYPE\_NONE, VTSS\_SFLOW\_TYPE\_RX, and VTSS\_SFLOW\_TYPE\_TX are possible.

Jaguar1 + Jaguar2: Note: `sflow_type`'s RX and TX enumerations are only reliable if ANA\_AC:PS\_COMMON:PS\_COMMON\_CFG.SFLOW\_SMPL\_ID\_IN\_STAMP\_ENA is set to 1. However, `sflow_type == VTSS_SFLOW_TYPE_NONE` is always reliable.

Validity:

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

Definition at line 1346 of file vtss\_packet\_api.h.

#### 6.74.2.18 sflow\_port\_no

`vtss_port_no_t vtss_packet_rx_info_t::sflow_port_no`

sFlow port. Indicates the logical sFlow Rx or Tx port number that caused this frame to be copied to the CPU. Only valid if `sflow_type` != VTSS\_SFLOW\_TYPE\_NONE.

Jaguar1 + Jaguar2: Note: This is only reliable if ANA\_AC:PS\_COMMON:PS\_COMMON\_CFG.SFLOW\_SMPL\_I $\leftarrow$ D\_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`.

#### 6.74.2.19 oam\_info

`u64 vtss_packet_rx_info_t::oam_info`

Various un-decodable OAM info. Decoding of the OAM info field from the extraction header requires accompanying frame info, and is therefore saved as an opaque type, letting it be up to the application to decode it based on description in the datasheet.

Serval: This corresponds to the contents of the REW\_VAL field in the extraction header. `oam_info_decoded` = TRUE when `REW_OP[2:0] == 4`. Only the 32 lsbits of `oam_info` are used.

Validity:

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

Definition at line 1393 of file `vtss_packet_api.h`.

#### 6.74.2.20 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`.

### 6.74.2.21 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.

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

- `vtss_api/include/vtss_packet_api.h`

## 6.75 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`

### 6.75.1 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.

## 6.75.2 Field Documentation

### 6.75.2.1 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.

### 6.75.2.2 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.

### 6.75.2.3 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.

### 6.75.2.4 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.

### 6.75.2.5 fcs

`u32 vtss_packet_rx_meta_t::fcs`

Frame checksum.

On some architectures, the frame's FCS/CRC may be used to hold additional side-band information about the frame itself.

To be able to decode this information, the caller must therefore find the end of the frame and extract the 32-bit FCS and place it here.

**Required to be set?**

```
Luton26: N
Jaguar1: Y (sflow-info or timestamp)
Serval: N
Jaguar2: Y (sfflow-info)
Serval2: Y (sfflow-info)
ServalT: Y (sfflow-info)
```

Definition at line 851 of file `vtss_packet_api.h`.

### 6.75.2.6 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`.

### 6.75.2.7 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](#)

## 6.76 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]

### 6.76.1 Detailed Description

Packet registration per port.

Definition at line 764 of file types.h.

## 6.76.2 Field Documentation

### 6.76.2.1 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.

### 6.76.2.2 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`

## 6.77 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`

### 6.77.1 Detailed Description

CPU Rx queue configuration.

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

### 6.77.2 Field Documentation

### 6.77.2.1 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.

### 6.77.2.2 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`

## 6.78 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`

### 6.78.1 Detailed Description

CPU Rx queue map.

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

## 6.78.2 Field Documentation

### 6.78.2.1 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.

### 6.78.2.2 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.

### 6.78.2.3 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.

### 6.78.2.4 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.

### 6.78.2.5 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.

### 6.78.2.6 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`.

### 6.78.2.7 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`.

### 6.78.2.8 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`.

### 6.78.2.9 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`

## 6.79 **vtss\_packet\_rx\_queue\_npi\_conf\_t** Struct Reference

CPU Rx queue NPI configuration.

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

## Data Fields

- `BOOL enable`

### 6.79.1 Detailed Description

CPU Rx queue NPI configuration.

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

### 6.79.2 Field Documentation

#### 6.79.2.1 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`

## 6.80 `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`

### 6.80.1 Detailed Description

CPU Rx packet registration.

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

## 6.80.2 Field Documentation

### 6.80.2.1 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.

### 6.80.2.2 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.

### 6.80.2.3 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.

### 6.80.2.4 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.

### 6.80.2.5 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`

## 6.81 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]`

### 6.81.1 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`.

### 6.81.2 Field Documentation

#### 6.81.2.1 length

`u32 vtss_packet_tx_ifh_t::length`

Length of compiled IFH (in bytes)

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

### 6.81.2.2 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

## 6.82 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\\_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](#)

### 6.82.1 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.

## 6.82.2 Field Documentation

### 6.82.2.1 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`.

### 6.82.2.2 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.

#### 6.82.2.3 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.

#### 6.82.2.4 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.

#### 6.82.2.5 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.

## 6.82.2.6 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.

## 6.82.2.7 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.

### 6.82.2.8 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.

### 6.82.2.9 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.

### 6.82.2.10 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.

### 6.82.2.11 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.

### 6.82.2.12 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`.

### 6.82.2.13 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`.

## 6.82.2.14 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.

## 6.82.2.15 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.

### 6.82.2.16 pdu\_offset

```
u32 vtss_packet_tx_info_t::pdu_offset
```

PDU offset in 8 bit word counts. Used in ptp-action's to indicate the start of the PTP PDU.

Validity: A F

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

Definition at line 1933 of file vtss\_packet\_api.h.

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

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

## 6.83 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`

### 6.83.1 Detailed Description

PHY auto negotiation advertisement.

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

## 6.83.2 Field Documentation

### 6.83.2.1 speed\_10m\_hdx

`BOOL vtss_phy_aneg_t::speed_10m_hdx`

10Mbps, half duplex

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

### 6.83.2.2 speed\_10m\_fdx

`BOOL vtss_phy_aneg_t::speed_10m_fdx`

10Mbps, full duplex

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

### 6.83.2.3 speed\_100m\_hdx

`BOOL vtss_phy_aneg_t::speed_100m_hdx`

100Mbps, half duplex

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

### 6.83.2.4 speed\_100m\_fdx

`BOOL vtss_phy_aneg_t::speed_100m_fdx`

100Mbps, full duplex

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

### 6.83.2.5 speed\_1g\_fdx

`BOOL vtss_phy_aneg_t::speed_1g_fdx`

1000Mbps, full duplex

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

### 6.83.2.6 speed\_1g\_hdx

`BOOL vtss_phy_aneg_t::speed_1g_hdx`

1000Mbps, full duplex

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

### 6.83.2.7 symmetric\_pause

`BOOL vtss_phy_aneg_t::symmetric_pause`

Symmetric pause

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

### 6.83.2.8 asymmetric\_pause

`BOOL vtss_phy_aneg_t::asymmetric_pause`

Asymmetric pause

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

### 6.83.2.9 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`

## 6.84 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`

### 6.84.1 Detailed Description

PHY clock configuration.

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

### 6.84.2 Field Documentation

#### 6.84.2.1 src

`vtss_phy_clk_source_t vtss_phy_clock_conf_t::src`

Clock source

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

#### 6.84.2.2 freq

`vtss_phy_freq_t vtss_phy_clock_conf_t::freq`

Clock frequency

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

#### 6.84.2.3 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`

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

PHY 1G configuration.

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

### Data Fields

- struct {  
    BOOL cfg  
    BOOL val  
} master

### 6.85.1 Detailed Description

PHY 1G configuration.

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

### 6.85.2 Field Documentation

#### 6.85.2.1 cfg

```
BOOL vtss_phy_conf_1g_t::cfg
```

Manual Master/Slave Config. 1=enabled

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

#### 6.85.2.2 val

```
BOOL vtss_phy_conf_1g_t::val
```

Master/Slave Config value, 1=Master

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

### 6.85.2.3 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](#)

## 6.86 vtss\_phy\_conf\_t Struct Reference

PHY configuration.

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

### Data Fields

- [vtss\\_phy\\_mode\\_t mode](#)
- [vtss\\_phy\\_forced\\_t forced](#)
- [vtss\\_phy\\_aneg\\_t aneg](#)
- [vtss\\_phy\\_mdi\\_t mdi](#)
- [vtss\\_phy\\_fast\\_link\\_fail\\_t flf](#)
- [vtss\\_phy\\_sigdet\\_polarity\\_t sigdet](#)
- [vtss\\_phy\\_unidirectional\\_t unidir](#)
- [vtss\\_phy\\_mac\\_serdes\\_pcs\\_ctrl\\_t mac\\_if\\_pcs](#)
- [vtss\\_phy\\_media\\_serdes\\_pcs\\_ctrl\\_t media\\_if\\_pcs](#)
- [vtss\\_phy\\_media\\_force\\_ams\\_sel\\_t force\\_ams\\_sel](#)
- [BOOL skip\\_coma](#)

### 6.86.1 Detailed Description

PHY configuration.

Definition at line 395 of file [vtss\\_phy\\_api.h](#).

### 6.86.2 Field Documentation

#### 6.86.2.1 mode

```
vtss_phy_mode_t vtss_phy_conf_t::mode
```

PHY mode

Definition at line 396 of file [vtss\\_phy\\_api.h](#).

### 6.86.2.2 forced

`vtss_phy_forced_t vtss_phy_conf_t::forced`

Forced mode configuration

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

### 6.86.2.3 aneg

`vtss_phy_aneg_t vtss_phy_conf_t::aneg`

Auto-negotiation mode configuration

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

### 6.86.2.4 mdi

`vtss_phy_mdi_t vtss_phy_conf_t::mdi`

Cu cable MDI (Crossed cable / normal cable)

Definition at line 399 of file vtss\_phy\_api.h.

### 6.86.2.5 flf

`vtss_phy_fast_link_fail_t vtss_phy_conf_t::flf`

Fast link failure configuration

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

### 6.86.2.6 sigdet

`vtss_phy_sigdet_polarity_t vtss_phy_conf_t::sigdet`

Sigdet pin polarity configuration

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

### 6.86.2.7 unidir

`vtss_phy_unidirectional_t` `vtss_phy_conf_t::unidir`

Unidirectional Configuration

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

### 6.86.2.8 mac\_if\_pcs

`vtss_phy_mac_serdes_pcs_cntl_t` `vtss_phy_conf_t::mac_if_pcs`

PHY MAC SerDes PCS Control (Reg16E3)

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

### 6.86.2.9 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 404 of file `vtss_phy_api.h`.

### 6.86.2.10 force\_ams\_sel

`vtss_phy_media_force_ams_sel_t` `vtss_phy_conf_t::force_ams_sel`

PHY Media AMS Force Selection

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

### 6.86.2.11 skip\_coma

`BOOL` `vtss_phy_conf_t::skip_coma`

PHY COMA Mode - Automatically apply COMA Config or SKIP

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

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

- `vtss_api/include/vtss_phy_api.h`

## 6.87 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`

#### 6.87.1 Detailed Description

EEE configuration.

Definition at line 987 of file vtss\_phy\_api.h.

#### 6.87.2 Field Documentation

##### 6.87.2.1 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.

##### 6.87.2.2 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](#)

## 6.88 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`

### 6.88.1 Detailed Description

enhanced LED control

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

### 6.88.2 Field Documentation

#### 6.88.2.1 `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`.

#### 6.88.2.2 `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`.

#### 6.88.2.3 `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`.

#### 6.88.2.4 ser\_led\_select

```
u8 vtss_phy_enhanced_led_control_t::ser_led_select
```

The number of LEDs the hardware board supports for each PHY, 0x00 = 4 LEDs, 0x01 = 3 LEDs, 0x2 = 2 LEDs, 0x3 = 1 LED

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

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

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

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

PHY forced mode configuration.

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

### Data Fields

- [vtss\\_port\\_speed\\_t speed](#)
- [BOOL fdx](#)

#### 6.89.1 Detailed Description

PHY forced mode configuration.

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

#### 6.89.2 Field Documentation

##### 6.89.2.1 speed

```
vtss_port_speed_t vtss_phy_forced_t::speed
```

Speed

Definition at line 304 of file vtss\_phy\_api.h.

### 6.89.2.2 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`

## 6.90 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`

#### 6.90.1 Detailed Description

LED model selection.

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

#### 6.90.2 Field Documentation

##### 6.90.2.1 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.

### 6.90.2.2 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`

## 6.91 `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`

### 6.91.1 Detailed Description

1G Phy loopbacks

Definition at line 1384 of file `vtss_phy_api.h`.

### 6.91.2 Field Documentation

#### 6.91.2.1 `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`.

### 6.91.2.2 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.

### 6.91.2.3 connector\_enable

`BOOL vtss_phy_loopback_t::connector_enable`

Enable/Disable loopback connector loopback

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

### 6.91.2.4 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.

### 6.91.2.5 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.

### 6.91.2.6 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.

### 6.91.2.7 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`.

### 6.91.2.8 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`.

### 6.91.2.9 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`

## 6.92 vtss\_phy\_mac\_serd\_pcs\_cntl\_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_serd_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`

### 6.92.1 Detailed Description

PHY MAC SerDes PCS Control, Reg16E3.

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

### 6.92.2 Field Documentation

#### 6.92.2.1 disable

```
BOOL vtss_phy_mac_serdes_pcs_cntl_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.

#### 6.92.2.2 restart

```
BOOL vtss_phy_mac_serdes_pcs_cntl_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.

#### 6.92.2.3 pd\_enable

```
BOOL vtss_phy_mac_serdes_pcs_cntl_t::pd_enable
```

MAC i/f ANEG parallel detect enable

Definition at line 354 of file vtss\_phy\_api.h.

#### 6.92.2.4 aneg\_restart

```
BOOL vtss_phy_mac_serdes_pcs_cntl_t::aneg_restart
```

Restart MAC i/f ANEG

Definition at line 355 of file vtss\_phy\_api.h.

### 6.92.2.5 force\_adv\_ability

`BOOL vtss_phy_mac_serdes_pcs_ctrl_t::force_adv_ability`

Force adv. ability from Reg18E3

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

### 6.92.2.6 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`.

### 6.92.2.7 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`.

### 6.92.2.8 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`.

### 6.92.2.9 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`.

#### 6.92.2.10 serdes\_pol\_inv\_out

`BOOL vtss_phy_mac_serdes_pcs_cntl_t::serdes_pol_inv_out`

Invert SerDes Polarity at output of MAC

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

#### 6.92.2.11 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.

#### 6.92.2.12 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`

## 6.93 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`

### 6.93.1 Detailed Description

PHY MEDIA SerDes PCS Control, Reg23E3.

Definition at line 375 of file vtss\_phy\_api.h.

### 6.93.2 Field Documentation

#### 6.93.2.1 remote\_fault

```
vtss_phy_media_rem_fault_t vtss_phy_media_serd_pcs_cntl_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.

#### 6.93.2.2 aneg\_pd\_detect

```
BOOL vtss_phy_media_serd_pcs_cntl_t::aneg_pd_detect
```

SerDes MEDIA ANEG parallel detect enable

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

#### 6.93.2.3 force\_adv\_ability

```
BOOL vtss_phy_media_serd_pcs_cntl_t::force_adv_ability
```

Force adv. ability from Reg25E3

Definition at line 378 of file vtss\_phy\_api.h.

#### 6.93.2.4 serdes\_pol\_inv\_in

```
BOOL vtss_phy_media_serd_pcs_cntl_t::serdes_pol_inv_in
```

Invert SerDes Polarity at input of Media SerDes

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

### 6.93.2.5 serdes\_pol\_inv\_out

`BOOL vtss_phy_media_serd_pcs_cntl_t::serdes_pol_inv_out`

Invert SerDes Polarity at output of Media SerDes

Definition at line 380 of file `vtss_phy_api.h`.

### 6.93.2.6 inhibit\_odd\_start

`BOOL vtss_phy_media_serd_pcs_cntl_t::inhibit_odd_start`

Inhibit Media Odd-Start delay

Definition at line 381 of file `vtss_phy_api.h`.

### 6.93.2.7 force\_hls

`BOOL vtss_phy_media_serd_pcs_cntl_t::force_hls`

Forces 100BaseFX to Tx HSL continuously

Definition at line 382 of file `vtss_phy_api.h`.

### 6.93.2.8 force\_fefi

`BOOL vtss_phy_media_serd_pcs_cntl_t::force_fefi`

Forces 100BaseFX Far-End-Fault Indication

Definition at line 383 of file `vtss_phy_api.h`.

### 6.93.2.9 force\_fefi\_value

`BOOL vtss_phy_media_serd_pcs_cntl_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`

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

PHY power configuration.

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

### Data Fields

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

#### 6.94.1 Detailed Description

PHY power configuration.

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

#### 6.94.2 Field Documentation

##### 6.94.2.1 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](#)

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

PHY power status.

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

### Data Fields

- [u32 level](#)

### 6.95.1 Detailed Description

PHY power status.

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

### 6.95.2 Field Documentation

#### 6.95.2.1 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](#)

## 6.96 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](#)

### 6.96.1 Detailed Description

PHY reset structure.

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

## 6.96.2 Field Documentation

### 6.96.2.1 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`.

### 6.96.2.2 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`.

### 6.96.2.3 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`.

### 6.96.2.4 tbi

`vtss_phy_tbi_conf_t vtss_phy_reset_conf_t::tbi`

TBI setup

Definition at line 222 of file `vtss_phy_api.h`.

### 6.96.2.5 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`.

### 6.96.2.6 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`.

### 6.96.2.7 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`

## 6.97 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`

### 6.97.1 Detailed Description

PHY RGMII configuration.

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

### 6.97.2 Field Documentation

### 6.97.2.1 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.

### 6.97.2.2 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](#)

## 6.98 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

### 6.98.1 Detailed Description

Phy statistic information.

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

### 6.98.2 Field Documentation

### 6.98.2.1 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.

### 6.98.2.2 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.

### 6.98.2.3 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.

### 6.98.2.4 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.

### 6.98.2.5 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.

### 6.98.2.6 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.

### 6.98.2.7 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](#)

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

PHY 1G status.

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

### Data Fields

- [BOOL master\\_cfg\\_fault](#)
- [BOOL master](#)

### 6.99.1 Detailed Description

PHY 1G status.

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

### 6.99.2 Field Documentation

### 6.99.2.1 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`.

### 6.99.2.2 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`

## 6.100 **vtss\_phy\_tbi\_conf\_t** Struct Reference

PHY TBI configuration.

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

### Data Fields

- `BOOL aneg_enable`

### 6.100.1 Detailed Description

PHY TBI configuration.

Definition at line 200 of file `vtss_phy_api.h`.

### 6.100.2 Field Documentation

### 6.100.2.1 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`

## 6.101 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`

### 6.101.1 Detailed Description

Phy type information.

Definition at line 143 of file `vtss_phy_api.h`.

### 6.101.2 Field Documentation

#### 6.101.2.1 part\_number

`u16 vtss_phy_type_t::part_number`

Part number

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

### 6.101.2.2 revision

`u16 vtss_phy_type_t::revision`

Chip revision

Definition at line 146 of file vtss\_phy\_api.h.

### 6.101.2.3 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.

### 6.101.2.4 channel\_id

`u16 vtss_phy_type_t::channel_id`

Channel id

Definition at line 148 of file vtss\_phy\_api.h.

### 6.101.2.5 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.

### 6.101.2.6 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](#)

## 6.102 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]`

#### 6.102.1 Detailed Description

VeriPHY result.

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

#### 6.102.2 Field Documentation

##### 6.102.2.1 link

```
BOOL vtss_phy_veriphy_result_t::link
```

Link status

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

##### 6.102.2.2 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.

### 6.102.2.3 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`

## 6.103 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`

### 6.103.1 Detailed Description

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

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

### 6.103.2 Field Documentation

#### 6.103.2.1 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.

### 6.103.2.2 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.

### 6.103.2.3 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.

### 6.103.2.4 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.

### 6.103.2.5 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](#)

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

PI configuration.

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

## Data Fields

- `vtss_pi_width_t width`
- `BOOL use_extended_bus_cycle`
- `u32 cs_wait_ns`

### 6.104.1 Detailed Description

PI configuration.

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

### 6.104.2 Field Documentation

#### 6.104.2.1 width

`vtss_pi_width_t vtss_pi_conf_t::width`

Width

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

#### 6.104.2.2 use\_extended\_bus\_cycle

`BOOL vtss_pi_conf_t::use_extended_bus_cycle`

Use extended bus cycle for slow registers

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

#### 6.104.2.3 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`

## 6.105 vtss\_policer\_ext\_t Struct Reference

Policer Extensions.

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

### Data Fields

- `BOOL frame_rate`
- `BOOL flow_control`

#### 6.105.1 Detailed Description

Policer Extensions.

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

#### 6.105.2 Field Documentation

##### 6.105.2.1 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.

##### 6.105.2.2 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

## 6.106 vtss\_policer\_t Struct Reference

Policer.

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

## Data Fields

- [vtss\\_burst\\_level\\_t](#) level
- [vtss\\_bitrate\\_t](#) rate

### 6.106.1 Detailed Description

Policer.

Definition at line 185 of file [vtss\\_qos\\_api.h](#).

### 6.106.2 Field Documentation

#### 6.106.2.1 level

[vtss\\_burst\\_level\\_t](#) vtss\_policer\_t::level

Burst level

Definition at line 187 of file [vtss\\_qos\\_api.h](#).

#### 6.106.2.2 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](#)

## 6.107 **vtss\_port\_bridge\_counters\_t** Struct Reference

Port bridge counter structure (RFC 4188)

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

## Data Fields

- [vtss\\_port\\_counter\\_t](#) dot1dTpPortInDiscards

### 6.107.1 Detailed Description

Port bridge counter structure (RFC 4188)

Definition at line 201 of file port.h.

### 6.107.2 Field Documentation

#### 6.107.2.1 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](#)

## 6.108 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](#)

### 6.108.1 Detailed Description

Advertisement control data for Clause 37 aneg.

Definition at line 127 of file vtss\_port\_api.h.

## 6.108.2 Field Documentation

### 6.108.2.1 fdx

`BOOL vtss_port_clause_37_adv_t::fdx`

(FD)

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

### 6.108.2.2 hdx

`BOOL vtss_port_clause_37_adv_t::hdx`

(HD)

Definition at line 130 of file vtss\_port\_api.h.

### 6.108.2.3 symmetric\_pause

`BOOL vtss_port_clause_37_adv_t::symmetric_pause`

(PS1)

Definition at line 131 of file vtss\_port\_api.h.

### 6.108.2.4 asymmetric\_pause

`BOOL vtss_port_clause_37_adv_t::asymmetric_pause`

(PS2)

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

### 6.108.2.5 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.

### 6.108.2.6 acknowledge

`BOOL vtss_port_clause_37_adv_t::acknowledge`

(Ack)

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

### 6.108.2.7 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`

## 6.109 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`

### 6.109.1 Detailed Description

Auto-negotiation control parameter struct.

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

### 6.109.2 Field Documentation

## 6.109.2.1 enable

```
BOOL vtss_port_clause_37_control_t::enable
```

Enable of Autoneg

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

## 6.109.2.2 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](#)

## 6.110 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](#)

### 6.110.1 Detailed Description

Port configuration structure.

Definition at line 268 of file vtss\_port\_api.h.

### 6.110.2 Field Documentation

#### 6.110.2.1 if\_type

`vtss_port_interface_t` `vtss_port_conf_t::if_type`

Interface type

Definition at line 270 of file vtss\_port\_api.h.

#### 6.110.2.2 sd\_enable

`BOOL` `vtss_port_conf_t::sd_enable`

Signal detect enable

Definition at line 271 of file vtss\_port\_api.h.

#### 6.110.2.3 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.

#### 6.110.2.4 sd\_internal

`BOOL` `vtss_port_conf_t::sd_internal`

Internal signal detect selection

Definition at line 273 of file vtss\_port\_api.h.

#### 6.110.2.5 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.

#### 6.110.2.6 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.

#### 6.110.2.7 speed

`vtss_port_speed_t` `vtss_port_conf_t::speed`

Port speed

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

#### 6.110.2.8 fdx

`BOOL` `vtss_port_conf_t::fdx`

Full duplex mode

Definition at line 277 of file vtss\_port\_api.h.

#### 6.110.2.9 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.

### 6.110.2.10 max\_frame\_length

`u32 vtss_port_conf_t::max_frame_length`

Maximum frame length

Definition at line 279 of file vtss\_port\_api.h.

### 6.110.2.11 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.

### 6.110.2.12 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.

### 6.110.2.13 exc\_col\_cont

`BOOL vtss_port_conf_t::exc_col_cont`

Excessive collision continuation

Definition at line 282 of file vtss\_port\_api.h.

### 6.110.2.14 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.

6.110.2.15 `xaui_tx_lane_flip`

```
BOOL vtss_port_conf_t::xaui_tx_lane_flip
```

XauI Tx lane flip

Definition at line 284 of file `vtss_port_api.h`.

6.110.2.16 `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`.

6.110.2.17 `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`

## 6.111 `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`
- `vtss_port_evc_counters_t evc`

### 6.111.1 Detailed Description

Port counter structure.

Definition at line 220 of file port.h.

### 6.111.2 Field Documentation

#### 6.111.2.1 rmon

```
vtss_port_rmon_counters_t vtss_port_counters_t::rmon
```

RMON counters

Definition at line 222 of file port.h.

#### 6.111.2.2 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.

#### 6.111.2.3 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.

#### 6.111.2.4 bridge

```
vtss_port_bridge_counters_t vtss_port_counters_t::bridge
```

Bridge counters

Definition at line 227 of file port.h.

### 6.111.2.5 prop

`vtss_port_proprietary_counters_t vtss_port_counters_t::prop`

Proprietary counters

Definition at line 230 of file port.h.

### 6.111.2.6 evc

`vtss_port_evc_counters_t vtss_port_counters_t::evc`

EVC counters

Definition at line 233 of file port.h.

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

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

## 6.112 vtss\_port\_ethernet\_like\_counters\_t Struct Reference

Ethernet-like Interface counter structure (RFC 3635)

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

### Data Fields

- `vtss_port_counter_t dot3InPauseFrames`
- `vtss_port_counter_t dot3OutPauseFrames`

### 6.112.1 Detailed Description

Ethernet-like Interface counter structure (RFC 3635)

Definition at line 163 of file port.h.

### 6.112.2 Field Documentation

### 6.112.2.1 dot3InPauseFrames

`vtss_port_counter_t vtss_port_ethernet_like_counters_t::dot3InPauseFrames`

Rx pause

Definition at line 171 of file port.h.

### 6.112.2.2 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`

## 6.113 vtss\_port\_evc\_counters\_t Struct Reference

EVC counters.

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

### Data Fields

- `vtss_port_counter_t rx_green [VTSS_PRIOS]`
- `vtss_port_counter_t rx_yellow [VTSS_PRIOS]`
- `vtss_port_counter_t rx_red [VTSS_PRIOS]`
- `vtss_port_counter_t rx_green_discard [VTSS_PRIOS]`
- `vtss_port_counter_t rx_yellow_discard [VTSS_PRIOS]`
- `vtss_port_counter_t tx_green [VTSS_PRIOS]`
- `vtss_port_counter_t tx_yellow [VTSS_PRIOS]`

### 6.113.1 Detailed Description

EVC counters.

Definition at line 186 of file port.h.

### 6.113.2 Field Documentation

### 6.113.2.1 rx\_green

```
vtss_port_counter_t vtss_port_evc_counters_t::rx_green[VTSS_PRIOS]
```

Rx green frames

Definition at line 189 of file port.h.

### 6.113.2.2 rx\_yellow

```
vtss_port_counter_t vtss_port_evc_counters_t::rx_yellow[VTSS_PRIOS]
```

Rx yellow frames

Definition at line 190 of file port.h.

### 6.113.2.3 rx\_red

```
vtss_port_counter_t vtss_port_evc_counters_t::rx_red[VTSS_PRIOS]
```

Rx red frames

Definition at line 191 of file port.h.

### 6.113.2.4 rx\_green\_discard

```
vtss_port_counter_t vtss_port_evc_counters_t::rx_green_discard[VTSS_PRIOS]
```

Rx green discarded frames

Definition at line 192 of file port.h.

### 6.113.2.5 rx\_yellow\_discard

```
vtss_port_counter_t vtss_port_evc_counters_t::rx_yellow_discard[VTSS_PRIOS]
```

Rx yellow discarded frames

Definition at line 193 of file port.h.

### 6.113.2.6 tx\_green

```
vtss_port_counter_t vtss_port_evc_counters_t::tx_green[VTSS_PRIOS]
```

Tx green frames

Definition at line 194 of file port.h.

### 6.113.2.7 tx\_yellow

```
vtss_port_counter_t vtss_port_evc_counters_t::tx_yellow[VTSS_PRIOS]
```

Tx yellow frames

Definition at line 195 of file port.h.

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

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

## 6.114 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 \[VTSS\\_PRIOS\]](#)

### 6.114.1 Detailed Description

Flow control setup.

Definition at line 189 of file vtss\_port\_api.h.

### 6.114.2 Field Documentation

**6.114.2.1 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.

**6.114.2.2 generate**

```
BOOL vtss_port_flow_control_conf_t::generate
```

TRUE if 802.3x PAUSE frames should generated

Definition at line 195 of file vtss\_port\_api.h.

**6.114.2.3 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.

**6.114.2.4 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](#)

## 6.115 **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`

### 6.115.1 Detailed Description

Inter frame gap structure.

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

### 6.115.2 Field Documentation

#### 6.115.2.1 `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`.

#### 6.115.2.2 `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`.

#### 6.115.2.3 `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`

## 6.116 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](#)

### 6.116.1 Detailed Description

Interfaces Group counter structure (RFC 2863)

Definition at line 144 of file port.h.

### 6.116.2 Field Documentation

#### 6.116.2.1 ifInOctets

```
vtss_port_counter_t vtss_port_if_group_counters_t::ifInOctets
```

Rx octets

Definition at line 145 of file port.h.

#### 6.116.2.2 ifInUcastPkts

```
vtss_port_counter_t vtss_port_if_group_counters_t::ifInUcastPkts
```

Rx unicasts

Definition at line 146 of file port.h.

### 6.116.2.3 ifInMulticastPkts

`vtss_port_counter_t vtss_port_if_group_counters_t::ifInMulticastPkts`

Rx multicasts

Definition at line 147 of file port.h.

### 6.116.2.4 ifInBroadcastPkts

`vtss_port_counter_t vtss_port_if_group_counters_t::ifInBroadcastPkts`

Rx broadcasts

Definition at line 148 of file port.h.

### 6.116.2.5 ifInNUcastPkts

`vtss_port_counter_t vtss_port_if_group_counters_t::ifInNUcastPkts`

Rx non-unicasts

Definition at line 149 of file port.h.

### 6.116.2.6 ifInDiscards

`vtss_port_counter_t vtss_port_if_group_counters_t::ifInDiscards`

Rx discards

Definition at line 150 of file port.h.

### 6.116.2.7 ifInErrors

`vtss_port_counter_t vtss_port_if_group_counters_t::ifInErrors`

Rx errors

Definition at line 151 of file port.h.

**6.116.2.8 ifOutOctets**

```
vtss_port_counter_t vtss_port_if_group_counters_t::ifOutOctets
```

Tx octets

Definition at line 153 of file port.h.

**6.116.2.9 ifOutUcastPkts**

```
vtss_port_counter_t vtss_port_if_group_counters_t::ifOutUcastPkts
```

Tx unicasts

Definition at line 154 of file port.h.

**6.116.2.10 ifOutMulticastPkts**

```
vtss_port_counter_t vtss_port_if_group_counters_t::ifOutMulticastPkts
```

Tx multicasts

Definition at line 155 of file port.h.

**6.116.2.11 ifOutBroadcastPkts**

```
vtss_port_counter_t vtss_port_if_group_counters_t::ifOutBroadcastPkts
```

Tx broadcasts

Definition at line 156 of file port.h.

**6.116.2.12 ifOutNUcastPkts**

```
vtss_port_counter_t vtss_port_if_group_counters_t::ifOutNUcastPkts
```

Tx non-unicasts

Definition at line 157 of file port.h.

### 6.116.2.13 ifOutDiscards

```
vtss_port_counter_t vtss_port_if_group_counters_t::ifOutDiscards
```

Tx discards

Definition at line 158 of file port.h.

### 6.116.2.14 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](#)

## 6.117 vtss\_port\_map\_t Struct Reference

Port map structure.

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

### Data Fields

- [i32 chip\\_port](#)
- [vtss\\_chip\\_no\\_t chip\\_no](#)
- [vtss\\_miim\\_controller\\_t miim\\_controller](#)
- [u8 miim\\_addr](#)
- [vtss\\_chip\\_no\\_t miim\\_chip\\_no](#)

### 6.117.1 Detailed Description

Port map structure.

Definition at line 72 of file vtss\_port\_api.h.

### 6.117.2 Field Documentation

### 6.117.2.1 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.

### 6.117.2.2 chip\_no

`vtss_chip_no_t vtss_port_map_t::chip_no`

MII management chip number, multi-chip targets

Definition at line 75 of file vtss\_port\_api.h.

### 6.117.2.3 miim\_controller

`vtss_miim_controller_t vtss_port_map_t::miim_controller`

MII management controller

Definition at line 80 of file vtss\_port\_api.h.

### 6.117.2.4 miim\_addr

`u8 vtss_port_map_t::miim_addr`

PHY address, ignored for VTSS\_MIIM\_CONTROLLER\_NONE

Definition at line 81 of file vtss\_port\_api.h.

### 6.117.2.5 miim\_chip\_no

`vtss_chip_no_t vtss_port_map_t::miim_chip_no`

MII management chip number, multi-chip targets

Definition at line 82 of file vtss\_port\_api.h.

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

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

## 6.118 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]`

#### 6.118.1 Detailed Description

Port proprietary counter structure.

Definition at line 207 of file port.h.

#### 6.118.2 Field Documentation

##### 6.118.2.1 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.

##### 6.118.2.2 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`

## 6.119 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`

### 6.119.1 Detailed Description

RMON counter structure (RFC 2819)

Definition at line 109 of file port.h.

### 6.119.2 Field Documentation

#### 6.119.2.1 rx\_etherStatsDropEvents

```
vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsDropEvents
```

Rx drop events

Definition at line 110 of file port.h.

### 6.119.2.2 rx\_etherStatsOctets

`vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsOctets`

Rx octets

Definition at line 111 of file port.h.

### 6.119.2.3 rx\_etherStatsPkts

`vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsPkts`

Rx packets

Definition at line 112 of file port.h.

### 6.119.2.4 rx\_etherStatsBroadcastPkts

`vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsBroadcastPkts`

Rx broadcasts

Definition at line 113 of file port.h.

### 6.119.2.5 rx\_etherStatsMulticastPkts

`vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsMulticastPkts`

Rx multicasts

Definition at line 114 of file port.h.

### 6.119.2.6 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.

**6.119.2.7 rx\_etherStatsUndersizePkts**

```
vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsUndersizePkts
```

Rx undersize packets

Definition at line 116 of file port.h.

**6.119.2.8 rx\_etherStatsOversizePkts**

```
vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsOversizePkts
```

Rx oversize packets

Definition at line 117 of file port.h.

**6.119.2.9 rx\_etherStatsFragments**

```
vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsFragments
```

Rx fragments

Definition at line 118 of file port.h.

**6.119.2.10 rx\_etherStatsJabbers**

```
vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsJabbers
```

Rx jabbers

Definition at line 119 of file port.h.

**6.119.2.11 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.

**6.119.2.12 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.

**6.119.2.13 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.

**6.119.2.14 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.

**6.119.2.15 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.

**6.119.2.16 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.

**6.119.2.17 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.

**6.119.2.18 tx\_etherStatsDropEvents**

```
vtss_port_counter_t vtss_port_rmon_counters_t::tx_etherStatsDropEvents
```

Tx drop events

Definition at line 128 of file port.h.

**6.119.2.19 tx\_etherStatsOctets**

```
vtss_port_counter_t vtss_port_rmon_counters_t::tx_etherStatsOctets
```

Tx octets

Definition at line 129 of file port.h.

**6.119.2.20 tx\_etherStatsPkts**

```
vtss_port_counter_t vtss_port_rmon_counters_t::tx_etherStatsPkts
```

Tx packets

Definition at line 130 of file port.h.

**6.119.2.21 tx\_etherStatsBroadcastPkts**

```
vtss_port_counter_t vtss_port_rmon_counters_t::tx_etherStatsBroadcastPkts
```

Tx broadcasts

Definition at line 131 of file port.h.

**6.119.2.22 tx\_etherStatsMulticastPkts**

```
vtss_port_counter_t vtss_port_rmon_counters_t::tx_etherStatsMulticastPkts
```

Tx multicasts

Definition at line 132 of file port.h.

**6.119.2.23 tx\_etherStatsCollisions**

```
vtss_port_counter_t vtss_port_rmon_counters_t::tx_etherStatsCollisions
```

Tx collisions

Definition at line 133 of file port.h.

**6.119.2.24 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.

**6.119.2.25 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.

**6.119.2.26 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.

**6.119.2.27 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.

**6.119.2.28 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.

**6.119.2.29 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.

**6.119.2.30 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](#)

## 6.120 vtss\_port\_serdes\_conf\_t Struct Reference

SFI Serdes configuration.

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

## Data Fields

- `BOOL sfp_dac`

### 6.120.1 Detailed Description

SFI Serdes configuration.

Definition at line 261 of file `vtss_port_api.h`.

### 6.120.2 Field Documentation

#### 6.120.2.1 `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`

## 6.121 `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`

### 6.121.1 Detailed Description

Advertisement control data for SGMII aneg.

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

## 6.121.2 Field Documentation

### 6.121.2.1 link

`BOOL vtss_port_sgmii_aneg_t::link`

LP link status

Definition at line 141 of file `vtss_port_api.h`.

### 6.121.2.2 fdx

`BOOL vtss_port_sgmii_aneg_t::fdx`

FD

Definition at line 142 of file `vtss_port_api.h`.

### 6.121.2.3 hdx

`BOOL vtss_port_sgmii_aneg_t::hdx`

HD

Definition at line 143 of file `vtss_port_api.h`.

### 6.121.2.4 speed\_10M

`BOOL vtss_port_sgmii_aneg_t::speed_10M`

speed 10 advertised

Definition at line 144 of file `vtss_port_api.h`.

### 6.121.2.5 speed\_100M

`BOOL vtss_port_sgmii_aneg_t::speed_100M`

speed 100 advertised

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

### 6.121.2.6 speed\_1G

`BOOL vtss_port_sgmii_aneg_t::speed_1G`

speed 1G advertised

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

### 6.121.2.7 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`

## 6.122 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`

### 6.122.1 Detailed Description

Port status parameter struct.

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

## 6.122.2 Field Documentation

### 6.122.2.1 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.

### 6.122.2.2 link

`BOOL` `vtss_port_status_t::link`

Link is up. Remaining fields only valid if TRUE

Definition at line 298 of file port.h.

### 6.122.2.3 speed

`vtss_port_speed_t` `vtss_port_status_t::speed`

Speed

Definition at line 299 of file port.h.

### 6.122.2.4 fdx

`BOOL` `vtss_port_status_t::fdx`

Full duplex

Definition at line 300 of file port.h.

### 6.122.2.5 remote\_fault

`BOOL` `vtss_port_status_t::remote_fault`

Remote fault signalled

Definition at line 301 of file port.h.

### 6.122.2.6 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.

### 6.122.2.7 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.

### 6.122.2.8 aneg

`vtss_aneg_t vtss_port_status_t::aneg`

Auto negotiation result

Definition at line 307 of file port.h.

### 6.122.2.9 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.

### 6.122.2.10 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.

### 6.122.2.11 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`

## 6.123 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`

### 6.123.1 Detailed Description

QCE action.

Definition at line 566 of file vtss\_qos\_api.h.

### 6.123.2 Field Documentation

#### 6.123.2.1 prio\_enable

`BOOL vtss_qce_action_t::prio_enable`

Enable priority classification

Definition at line 568 of file vtss\_qos\_api.h.

### 6.123.2.2 prio

`vtss_prio_t vtss_qce_action_t::prio`

Priority value

Definition at line 569 of file vtss\_qos\_api.h.

### 6.123.2.3 dp\_enable

`BOOL vtss_qce_action_t::dp_enable`

Enable DP classification

Definition at line 570 of file vtss\_qos\_api.h.

### 6.123.2.4 dp

`vtss_dp_level_t vtss_qce_action_t::dp`

DP value

Definition at line 571 of file vtss\_qos\_api.h.

### 6.123.2.5 dscp\_enable

`BOOL vtss_qce_action_t::dscp_enable`

Enable DSCP classification

Definition at line 572 of file vtss\_qos\_api.h.

### 6.123.2.6 dscp

`vtss_dscp_t vtss_qce_action_t::dscp`

DSCP value

Definition at line 573 of file vtss\_qos\_api.h.

### 6.123.2.7 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.

### 6.123.2.8 pcp

`vtss_tagprio_t vtss_qce_action_t::pcp`

PCP value

Definition at line 576 of file vtss\_qos\_api.h.

### 6.123.2.9 dei

`vtss_dei_t vtss_qce_action_t::dei`

DEI value

Definition at line 577 of file vtss\_qos\_api.h.

### 6.123.2.10 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.

### 6.123.2.11 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](#)
-

## 6.124 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`

#### 6.124.1 Detailed Description

Frame data for VTSS\_QCE\_TYPEETYPE.

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

#### 6.124.2 Field Documentation

##### 6.124.2.1 etype

```
vtss_vcap_u16_t vtss_qce_frame_etype_t::etype
```

Ethernet Type value

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

##### 6.124.2.2 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

## 6.125 vtss\_qce\_frame\_ipv4\_t Struct Reference

Frame data for VTSS\_QCE\_TYPE\_IPV4.

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

## Data Fields

- `vtss_vcap_bit_t` fragment
- `vtss_vcap_vr_t` dscp
- `vtss_vcap_u8_t` proto
- `vtss_vcap_ip_t` sip
- `vtss_vcap_vr_t` sport
- `vtss_vcap_vr_t` dport

### 6.125.1 Detailed Description

Frame data for VTSS\_QCE\_TYPE\_IPV4.

Definition at line 513 of file vtss\_qos\_api.h.

### 6.125.2 Field Documentation

#### 6.125.2.1 fragment

`vtss_vcap_bit_t` vtss\_qce\_frame\_ipv4\_t::fragment

Fragment

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

#### 6.125.2.2 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.

#### 6.125.2.3 proto

`vtss_vcap_u8_t` vtss\_qce\_frame\_ipv4\_t::proto

Protocol

Definition at line 517 of file vtss\_qos\_api.h.

#### 6.125.2.4 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.

#### 6.125.2.5 sport

`vtss_vcap_vr_t vtss_qce_frame_ipv4_t::sport`

UDP/TCP: Source port - Serval: key\_type = normal, ip\_addr and mac\_ip\_addr

Definition at line 522 of file vtss\_qos\_api.h.

#### 6.125.2.6 dport

`vtss_vcap_vr_t vtss_qce_frame_ipv4_t::dport`

UDP/TCP: Destination port - Serval: key\_type = double\_tag, ip\_addr and mac\_ip\_addr

Definition at line 523 of file vtss\_qos\_api.h.

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

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

## 6.126 vtss\_qce\_frame\_ipv6\_t Struct Reference

Frame data for VTSS\_QCE\_TYPE\_IPV6.

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

### Data Fields

- `vtss_vcap_vr_t dscp`
- `vtss_vcap_u8_t proto`
- `vtss_vcap_u128_t sip`
- `vtss_vcap_vr_t sport`
- `vtss_vcap_vr_t dport`

## 6.126.1 Detailed Description

Frame data for VTSS\_QCE\_TYPE\_IPV6.

Definition at line 527 of file vtss\_qos\_api.h.

## 6.126.2 Field Documentation

### 6.126.2.1 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.

### 6.126.2.2 proto

`vtss_vcap_u8_t` vtss\_qce\_frame\_ipv6\_t::proto

Protocol

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

### 6.126.2.3 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.

### 6.126.2.4 sport

`vtss_vcap_vr_t` vtss\_qce\_frame\_ipv6\_t::sport

UDP/TCP: Source port - Serval: key\_type = normal, ip\_addr and mac\_ip\_addr

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

### 6.126.2.5 dport

`vtss_vcap_vr_t vtss_qce_frame_ipv6_t::dport`

UDP/TCP: Destination port - Serval: key\_type = double\_tag, ip\_addr and mac\_ip\_addr

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

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

- `vtss_api/include/vtss_qos_api.h`

## 6.127 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`

### 6.127.1 Detailed Description

Frame data for VTSS\_QCE\_TYPE LLC.

Definition at line 501 of file vtss\_qos\_api.h.

### 6.127.2 Field Documentation

#### 6.127.2.1 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`

## 6.128 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

#### 6.128.1 Detailed Description

Frame data for VTSS\_QCE\_TYPE\_SNAP.

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

#### 6.128.2 Field Documentation

##### 6.128.2.1 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

## 6.129 vtss\_qce\_key\_t Struct Reference

QCE key.

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

### Data Fields

- `BOOL` port\_list [VTSS\_PORT\_ARRAY\_SIZE]
- `vtss_qce_mac_t` mac
- `vtss_qce_tag_t` tag
- `vtss_qce_type_t` type
- union {
  - `vtss_qce_frame_etype_t` etype
  - `vtss_qce_frame_llc_t` llc
  - `vtss_qce_frame_snap_t` snap
  - `vtss_qce_frame_ipv4_t` ipv4
  - `vtss_qce_frame_ipv6_t` ipv6}
- `vtss_qce_frame_t` frame

### 6.129.1 Detailed Description

QCE key.

Definition at line 542 of file vtss\_qos\_api.h.

### 6.129.2 Field Documentation

#### 6.129.2.1 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.

#### 6.129.2.2 mac

```
vtss_qce_mac_t vtss_qce_key_t::mac
```

MAC

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

#### 6.129.2.3 tag

```
vtss_qce_tag_t vtss_qce_key_t::tag
```

Tag

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

#### 6.129.2.4 type

```
vtss_qce_type_t vtss_qce_key_t::type
```

Frame type

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

**6.129.2.5 etype**

```
vtss_qce_frame_etype_t vtss_qce_key_t::etype
```

VTSS\_QCE\_TYPE\_ETYPE

Definition at line 555 of file vtss\_qos\_api.h.

**6.129.2.6 llc**

```
vtss_qce_frame_llc_t vtss_qce_key_t::llc
```

VTSS\_QCE\_TYPE\_LLCC

Definition at line 556 of file vtss\_qos\_api.h.

**6.129.2.7 snap**

```
vtss_qce_frame_snap_t vtss_qce_key_t::snap
```

VTSS\_QCE\_TYPE\_SNAP

Definition at line 557 of file vtss\_qos\_api.h.

**6.129.2.8 ipv4**

```
vtss_qce_frame_ipv4_t vtss_qce_key_t::ipv4
```

VTSS\_QCE\_TYPE\_IPV4

Definition at line 558 of file vtss\_qos\_api.h.

**6.129.2.9 ipv6**

```
vtss_qce_frame_ipv6_t vtss_qce_key_t::ipv6
```

VTSS\_QCE\_TYPE\_IPV6

Definition at line 559 of file vtss\_qos\_api.h.

### 6.129.2.10 frame

```
union { ... } vtss_qce_key_t::frame
```

Frame type specific data

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

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

## 6.130 vtss\_qce\_mac\_t Struct Reference

QCE MAC information.

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

### Data Fields

- [vtss\\_vcap\\_bit\\_t dmac\\_mc](#)
- [vtss\\_vcap\\_bit\\_t dmac\\_bc](#)
- [vtss\\_vcap\\_u48\\_t smac](#)

### 6.130.1 Detailed Description

QCE MAC information.

Definition at line 471 of file [vtss\\_qos\\_api.h](#).

### 6.130.2 Field Documentation

#### 6.130.2.1 dmac\_mc

```
vtss\_vcap\_bit\_t vtss_qce_mac_t::dmac_mc
```

Multicast DMAC

Definition at line 473 of file [vtss\\_qos\\_api.h](#).

### 6.130.2.2 dmac\_bc

`vtss_vcap_bit_t` `vtss_qce_mac_t::dmac_bc`

Broadcast DMAC

Definition at line 474 of file `vtss_qos_api.h`.

### 6.130.2.3 smac

`vtss_vcap_u48_t` `vtss_qce_mac_t::smac`

SMAC - Only the 24 most significant bits (OUI) are supported on Jaguar1, rest are wildcards

Definition at line 478 of file `vtss_qos_api.h`.

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

- `vtss_api/include/vtss_qos_api.h`

## 6.131 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`

### 6.131.1 Detailed Description

QoS Control Entry.

Definition at line 588 of file `vtss_qos_api.h`.

### 6.131.2 Field Documentation

### 6.131.2.1 id

`vtss_qce_id_t` `vtss_qce_t::id`

Entry ID

Definition at line 590 of file `vtss_qos_api.h`.

### 6.131.2.2 key

`vtss_qce_key_t` `vtss_qce_t::key`

QCE key

Definition at line 591 of file `vtss_qos_api.h`.

### 6.131.2.3 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`

## 6.132 `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`

### 6.132.1 Detailed Description

QCE tag information.

Definition at line 482 of file vtss\_qos\_api.h.

### 6.132.2 Field Documentation

#### 6.132.2.1 vid

`vtss_vcap_vr_t` `vtss_qce_tag_t::vid`

VLAN ID (12 bit)

Definition at line 484 of file vtss\_qos\_api.h.

#### 6.132.2.2 pcp

`vtss_vcap_u8_t` `vtss_qce_tag_t::pcp`

PCP (3 bit)

Definition at line 485 of file vtss\_qos\_api.h.

#### 6.132.2.3 dei

`vtss_vcap_bit_t` `vtss_qce_tag_t::dei`

DEI

Definition at line 486 of file vtss\_qos\_api.h.

#### 6.132.2.4 tagged

`vtss_vcap_bit_t` `vtss_qce_tag_t::tagged`

Tagged/untagged frame

Definition at line 487 of file vtss\_qos\_api.h.

### 6.132.2.5 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`

## 6.133 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]`
- `BOOL dscp_remark [64]`
- `vtss_dscp_t dscp_translate_map [64]`
- `vtss_dscp_t dscp_remap [64]`
- `vtss_dscp_t dscp_remap_dp1 [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`

### 6.133.1 Detailed Description

All parameters below are defined per chip.

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

### 6.133.2 Field Documentation

### 6.133.2.1 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.

### 6.133.2.2 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.

### 6.133.2.3 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.

### 6.133.2.4 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.

### 6.133.2.5 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.

### 6.133.2.6 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.

### 6.133.2.7 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.

### 6.133.2.8 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.

### 6.133.2.9 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.

### 6.133.2.10 dscp\_remap\_dp1

```
vtss_dscp_t vtss_qos_conf_t::dscp_remap_dp1[64]
```

Egress: Remap one DSCP to another (DP aware and DP level = 1)

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

### 6.133.2.11 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.

### 6.133.2.12 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.

### 6.133.2.13 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.

### 6.133.2.14 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.

### 6.133.2.15 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.

### 6.133.2.16 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`.

### 6.133.2.17 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`.

### 6.133.2.18 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`.

### 6.133.2.19 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`.

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

- `vtss_api/include/vtss_qos_api.h`

## 6.134 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]`
- `BOOL excess_enable [VTSS_QUEUE_ARRAY_SIZE]`
- `vtss_prio_t default_prio`
- `vtss_tagprio_t usr_prio`
- `vtss_dp_level_t default_dpl`
- `vtss_dei_t default_dei`
- `BOOL tag_class_enable`
- `vtss_prio_t qos_class_map [VTSS_PCP_ARRAY_SIZE][VTSS_DEI_ARRAY_SIZE]`
- `vtss_dp_level_t dp_level_map [VTSS_PCP_ARRAY_SIZE][VTSS_DEI_ARRAY_SIZE]`
- `BOOL dscp_class_enable`
- `vtss_dscp_mode_t dscp_mode`
- `vtss_dscp_emode_t dscp_emode`
- `BOOL dscp_translate`
- `vtss_tag_remark_mode_t tag_remark_mode`
- `vtss_tagprio_t tag_default_pcp`
- `vtss_dei_t tag_default_dei`
- `vtss_tagprio_t tag_pcp_map [VTSS_PRIO_ARRAY_SIZE][2]`
- `vtss_dei_t tag_dei_map [VTSS_PRIO_ARRAY_SIZE][2]`
- `BOOL dwrr_enable`
- `vtss_pct_t queue_pct [VTSS_QUEUE_ARRAY_SIZE]`
- `BOOL dmac_dip`

### 6.134.1 Detailed Description

QoS setup per port.

Definition at line 344 of file vtss\_qos\_api.h.

### 6.134.2 Field Documentation

#### 6.134.2.1 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.

#### 6.134.2.2 **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.

#### 6.134.2.3 **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.

#### 6.134.2.4 **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.

#### 6.134.2.5 **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.

#### 6.134.2.6 **excess\_enable**

```
BOOL vtss_qos_port_conf_t::excess_enable[VTSS_QUEUE_ARRAY_SIZE]
```

Allow this queue to use excess bandwidth

Definition at line 365 of file vtss\_qos\_api.h.

#### 6.134.2.7 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.

#### 6.134.2.8 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.

#### 6.134.2.9 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.

#### 6.134.2.10 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.

#### 6.134.2.11 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.

#### 6.134.2.12 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.

#### 6.134.2.13 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.

#### 6.134.2.14 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.

#### 6.134.2.15 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.

#### 6.134.2.16 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.

**6.134.2.17 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.

**6.134.2.18 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.

**6.134.2.19 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.

**6.134.2.20 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.

**6.134.2.21 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.

#### 6.134.2.22 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.

#### 6.134.2.23 dwrr\_enable

```
BOOL vtss_qos_port_conf_t::dwrr_enable
```

Enable Weighted fairness queueing

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

#### 6.134.2.24 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.

#### 6.134.2.25 dmac\_dip

```
BOOL vtss_qos_port_conf_t::dmac_dip
```

Enable DMAC/DIP matching in QCLs (default SMAC/SIP)

Definition at line 408 of file vtss\_qos\_api.h.

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

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

### 6.135 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`

### 6.135.1 Detailed Description

Structure for Get PHY RC-PLL status.

Definition at line 1742 of file `vtss_phy_api.h`.

### 6.135.2 Field Documentation

#### 6.135.2.1 `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`.

#### 6.135.2.2 `cal_error`

```
u8 vtss_rcpll_status_t::cal_error
```

Calibration Error indication

Definition at line 1744 of file `vtss_phy_api.h`.

#### 6.135.2.3 `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`

## 6.136 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`

#### 6.136.1 Detailed Description

Restart status.

Definition at line 608 of file vtss\_init\_api.h.

#### 6.136.2 Field Documentation

##### 6.136.2.1 restart

```
vtss_restart_t vtss_restart_status_t::restart
```

Previous restart mode

Definition at line 609 of file vtss\_init\_api.h.

##### 6.136.2.2 prev\_version

```
vtss_version_t vtss_restart_status_t::prev_version
```

Previous API version

Definition at line 610 of file vtss\_init\_api.h.

## 6.136.2.3 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

## 6.137 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](#)

### 6.137.1 Detailed Description

Routing entry.

Definition at line 868 of file types.h.

### 6.137.2 Field Documentation

#### 6.137.2.1 type

```
vtss_routing_entry_type_t vtss_routing_entry_t::type
```

Type of route

Definition at line 871 of file types.h.

### 6.137.2.2 ipv4\_uc

`vtss_ipv4_uc_t` `vtss_routing_entry_t::ipv4_uc`

IPv6 unicast route

Definition at line 875 of file types.h.

### 6.137.2.3 ipv6\_uc

`vtss_ipv6_uc_t` `vtss_routing_entry_t::ipv6_uc`

IPv6 unicast route

Definition at line 878 of file types.h.

### 6.137.2.4 route

`union { ... } vtss_routing_entry_t::route`

Route

### 6.137.2.5 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`

## 6.138 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]`

### 6.138.1 Detailed Description

Structure for Wake-On-LAN Secure-On Password.

Definition at line 1664 of file vtss\_phy\_api.h.

### 6.138.2 Field Documentation

#### 6.138.2.1 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](#)

## 6.139 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](#)

### 6.139.1 Detailed Description

Serdes macro configuration.

Definition at line 363 of file vtss\_init\_api.h.

### 6.139.2 Field Documentation

### 6.139.2.1 serdes1g\_vdd

`vtss_vdd_t` `vtss_serdes_macro_conf_t::serdes1g_vdd`

Serdes1g supply

Definition at line 364 of file `vtss_init_api.h`.

### 6.139.2.2 serdes6g\_vdd

`vtss_vdd_t` `vtss_serdes_macro_conf_t::serdes6g_vdd`

Serdes6g supply

Definition at line 365 of file `vtss_init_api.h`.

### 6.139.2.3 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`.

### 6.139.2.4 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`

## 6.140 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`

### 6.140.1 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`.

### 6.140.2 Field Documentation

#### 6.140.2.1 `type`

`vtss_sflow_type_t vtss_sflow_port_conf_t::type`

Sample direction. Also used to turn off sampling.

Definition at line 1451 of file `vtss_l2_api.h`.

#### 6.140.2.2 `sampling_rate`

`u32 vtss_sflow_port_conf_t::sampling_rate`

A value of N means: sample on average 1 out of N frames. 0 disables sampling.

Definition at line 1452 of file `vtss_l2_api.h`.

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

- `vtss_api/include/vtss_l2_api.h`

## 6.141 **vtss\_sgpi\_conf\_t** Struct Reference

GPIO configuration for a group.

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

## Data Fields

- `vtss_sgpio_bmode_t bmode [2]`
- `u8 bit_count`
- `vtss_sgpio_port_conf_t port_conf [VTSS_SGPIO_PORTS]`

### 6.141.1 Detailed Description

SGPIO configuration for a group.

Definition at line 797 of file vtss\_misc\_api.h.

### 6.141.2 Field Documentation

#### 6.141.2.1 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.

#### 6.141.2.2 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.

#### 6.141.2.3 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

## 6.142 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]`

#### 6.142.1 Detailed Description

SGPIO port configuration.

Definition at line 789 of file vtss\_misc\_api.h.

#### 6.142.2 Field Documentation

##### 6.142.2.1 enabled

```
BOOL vtss_sgpio_port_conf_t::enabled
```

Port enabled/disabled

Definition at line 791 of file vtss\_misc\_api.h.

##### 6.142.2.2 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.

### 6.142.2.3 int\_pol\_high

```
BOOL vtss_sgpi_port_conf_t::int_pol_high[4]
```

SGPIO interrupt polarity for each bit - TRUE - Interrupt when SGPIO pin high, FALSE - Interrupt when SGPIO pin low.

Definition at line 793 of file vtss\_misc\_api.h.

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

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

## 6.143 vtss\_sgpi\_port\_data\_t Struct Reference

SGPIO read data for a port.

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

### Data Fields

- [BOOL value](#) [4]

### 6.143.1 Detailed Description

SGPIO read data for a port.

Definition at line 835 of file vtss\_misc\_api.h.

### 6.143.2 Field Documentation

#### 6.143.2.1 value

```
BOOL vtss_sgpi_port_data_t::value[4]
```

Data for each and bit

Definition at line 837 of file vtss\_misc\_api.h.

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

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

## 6.144 vtss\_shaper\_t Struct Reference

Shaper.

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

### Data Fields

- `vtss_burst_level_t level`
- `vtss_bitrate_t rate`

#### 6.144.1 Detailed Description

Shaper.

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

#### 6.144.2 Field Documentation

##### 6.144.2.1 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`.

##### 6.144.2.2 rate

```
vtss_bitrate_t vtss_shaper_t::rate
```

CIR (Committed Information Rate). Unit: kbps. Use `VTSS_BITRATE_DISABLED` to disable shaper

Definition at line 301 of file `vtss_qos_api.h`.

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

- `vtss_api/include/vtss_qos_api.h`

## 6.145 vtss\_sync\_clock\_in\_t Struct Reference

Struct containing configuration selecting the recovered input clock port, to be delivered to a selected output clock port.

```
#include <vtss_sync_api.h>
```

### Data Fields

- `vtss_port_no_t port_no`
- `BOOL squelsh`
- `BOOL enable`

#### 6.145.1 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`.

#### 6.145.2 Field Documentation

##### 6.145.2.1 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`.

##### 6.145.2.2 squelsh

```
BOOL vtss_sync_clock_in_t::squelsh
```

Enable/disable of automatic squelsh

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

### 6.145.2.3 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`

## 6.146 `vtss_sync_clock_out_t` Struct Reference

Struct containing configuration for a recovered clock output port.

```
#include <vtss_sync_api.h>
```

### Data Fields

- `vtss_sync_divider_t divider`
- `BOOL enable`

### 6.146.1 Detailed Description

Struct containing configuration for a recovered clock output port.

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

### 6.146.2 Field Documentation

#### 6.146.2.1 divider

`vtss_sync_divider_t vtss_sync_clock_out_t::divider`

Selection the clock division. This should be set to `VTSS_SYNC_DIVIDER_1` if recovered clock is comming from internal PHY

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

### 6.146.2.2 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`

## 6.147 `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`

### 6.147.1 Detailed Description

Tag Control Information (according to IEEE 802.1Q)

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

### 6.147.2 Field Documentation

#### 6.147.2.1 vid

`vtss_vid_t vtss_tci_t::vid`

VLAN ID

Definition at line 208 of file `vtss_packet_api.h`.

### 6.147.2.2 cfi

`BOOL vtss_tci_t::cfi`

Canonical Format Indicator

Definition at line 209 of file vtss\_packet\_api.h.

### 6.147.2.3 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](#)

## 6.148 vtss\_timeofday\_t Struct Reference

Time of day structure.

```
#include <vtss_os_ecos.h>
```

### Data Fields

- `u32 sec`
- `time_t sec`

### 6.148.1 Detailed Description

Time of day structure.

Definition at line 59 of file vtss\_os\_ecos.h.

### 6.148.2 Field Documentation

### 6.148.2.1 sec [1/2]

`u32 vtss_timeofday_t::sec`

Time of day in seconds

Definition at line 60 of file `vtss_os_ecos.h`.

### 6.148.2.2 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`

## 6.149 vtss\_timestamp\_t Struct Reference

Time stamp in seconds and nanoseconds.

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

### Data Fields

- `u16 sec_msb`
- `u32 seconds`
- `u32 nanoseconds`

### 6.149.1 Detailed Description

Time stamp in seconds and nanoseconds.

Definition at line 1212 of file `types.h`.

### 6.149.2 Field Documentation

#### 6.149.2.1 sec\_msb

`u16 vtss_timestamp_t::sec_msb`

Seconds msb

Definition at line 1213 of file types.h.

#### 6.149.2.2 seconds

`u32 vtss_timestamp_t::seconds`

Seconds

Definition at line 1214 of file types.h.

#### 6.149.2.3 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`

## 6.150 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]`

#### 6.150.1 Detailed Description

Trace group configuration.

Definition at line 97 of file vtss\_misc\_api.h.

## 6.150.2 Field Documentation

### 6.150.2.1 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](#)

## 6.151 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](#)

### 6.151.1 Detailed Description

external clock output configuration.

Definition at line 289 of file vtss\_ts\_api.h.

### 6.151.2 Field Documentation

#### 6.151.2.1 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.

### 6.151.2.2 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.

### 6.151.2.3 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`

## 6.152 vtss\_ts\_id\_t Struct Reference

Timestamp identifier.

```
#include <vtss_ts_api.h>
```

### Data Fields

- `u32 ts_id`

### 6.152.1 Detailed Description

Timestamp identifier.

Definition at line 527 of file vtss\_ts\_api.h.

### 6.152.2 Field Documentation

### 6.152.2.1 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](#)

## 6.153 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](#)

### 6.153.1 Detailed Description

Hardware timestamping format mode for internal ports.

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

### 6.153.2 Field Documentation

#### 6.153.2.1 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](#)

## 6.154 vtss\_ts\_operation\_mode\_t Struct Reference

Timestamp operation.

```
#include <vtss_ts_api.h>
```

### Data Fields

- [vtss\\_ts\\_mode\\_t mode](#)

#### 6.154.1 Detailed Description

Timestamp operation.

Definition at line 451 of file vtss\_ts\_api.h.

#### 6.154.2 Field Documentation

##### 6.154.2.1 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](#)

## 6.155 vtss\_ts\_timestamp\_alloc\_t Struct Reference

Timestamp allocation.

```
#include <vtss_ts_api.h>
```

### Data Fields

- [u64 port\\_mask](#)
- [void \\* context](#)
- [void\(\\* cb \)\(void \\*context, u32 port\\_no, vtss\\_ts\\_timestamp\\_t \\*ts\)](#)

### 6.155.1 Detailed Description

Timestamp allocation.

Definition at line 613 of file vtss\_ts\_api.h.

### 6.155.2 Field Documentation

#### 6.155.2.1 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.

#### 6.155.2.2 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.

#### 6.155.2.3 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](#)

## 6.156 vtss\_ts\_timestamp\_t Struct Reference

Timestamp structure.

```
#include <vtss_ts_api.h>
```

## Data Fields

- `u32 ts`
- `u32 id`
- `void * context`
- `BOOL ts_valid`

### 6.156.1 Detailed Description

Timestamp structure.

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

### 6.156.2 Field Documentation

#### 6.156.2.1 ts

`u32 vtss_ts_timestamp_t::ts`

Timestamp value

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

#### 6.156.2.2 id

`u32 vtss_ts_timestamp_t::id`

Timestamp identifier

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

#### 6.156.2.3 context

`void* vtss_ts_timestamp_t::context`

Application specific context

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

#### 6.156.2.4 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`

### 6.157 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`

#### 6.157.1 Detailed Description

VCAP IPv4 address value and mask.

Definition at line 968 of file `types.h`.

#### 6.157.2 Field Documentation

##### 6.157.2.1 value

`vtss_ip_t vtss_vcap_ip_t::value`

Value

Definition at line 970 of file `types.h`.

### 6.157.2.2 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`

## 6.158 vtss\_vcap\_u128\_t Struct Reference

VCAP 128 bit value and mask.

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

### Data Fields

- `u8 value [16]`
- `u8 mask [16]`

### 6.158.1 Detailed Description

VCAP 128 bit value and mask.

Definition at line 954 of file types.h.

### 6.158.2 Field Documentation

#### 6.158.2.1 value

`u8 vtss_vcap_u128_t::value[16]`

Value

Definition at line 956 of file types.h.

### 6.158.2.2 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](#)

## 6.159 vtss\_vcap\_u16\_t Struct Reference

VCAP 16 bit value and mask.

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

### Data Fields

- [u8 value \[2\]](#)
- [u8 mask \[2\]](#)

### 6.159.1 Detailed Description

VCAP 16 bit value and mask.

Definition at line 919 of file types.h.

### 6.159.2 Field Documentation

#### 6.159.2.1 value

`u8 vtss_vcap_u16_t::value[2]`

Value

Definition at line 921 of file types.h.

### 6.159.2.2 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](#)

## 6.160 vtss\_vcap\_u24\_t Struct Reference

VCAP 24 bit value and mask.

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

### Data Fields

- [u8 value \[3\]](#)
- [u8 mask \[3\]](#)

### 6.160.1 Detailed Description

VCAP 24 bit value and mask.

Definition at line 926 of file types.h.

### 6.160.2 Field Documentation

#### 6.160.2.1 value

`u8 vtss_vcap_u24_t::value [3]`

Value

Definition at line 928 of file types.h.

### 6.160.2.2 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](#)

## 6.161 vtss\_vcap\_u32\_t Struct Reference

VCAP 32 bit value and mask.

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

### Data Fields

- [u8 value \[4\]](#)
- [u8 mask \[4\]](#)

### 6.161.1 Detailed Description

VCAP 32 bit value and mask.

Definition at line 933 of file types.h.

### 6.161.2 Field Documentation

#### 6.161.2.1 value

`u8 vtss_vcap_u32_t::value[4]`

Value

Definition at line 935 of file types.h.

### 6.161.2.2 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](#)

## 6.162 vtss\_vcap\_u40\_t Struct Reference

VCAP 40 bit value and mask.

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

### Data Fields

- [u8 value \[5\]](#)
- [u8 mask \[5\]](#)

### 6.162.1 Detailed Description

VCAP 40 bit value and mask.

Definition at line 940 of file types.h.

### 6.162.2 Field Documentation

#### 6.162.2.1 value

`u8 vtss_vcap_u40_t::value[5]`

Value

Definition at line 942 of file types.h.

### 6.162.2.2 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](#)

## 6.163 vtss\_vcap\_u48\_t Struct Reference

VCAP 48 bit value and mask.

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

### Data Fields

- [u8 value \[6\]](#)
- [u8 mask \[6\]](#)

### 6.163.1 Detailed Description

VCAP 48 bit value and mask.

Definition at line 947 of file types.h.

### 6.163.2 Field Documentation

#### 6.163.2.1 value

`u8 vtss_vcap_u48_t::value[6]`

Value

Definition at line 949 of file types.h.

### 6.163.2.2 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](#)

## 6.164 vtss\_vcap\_u8\_t Struct Reference

VCAP 8 bit value and mask.

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

### Data Fields

- [u8 value](#)
- [u8 mask](#)

### 6.164.1 Detailed Description

VCAP 8 bit value and mask.

Definition at line 912 of file types.h.

### 6.164.2 Field Documentation

#### 6.164.2.1 value

`u8 vtss_vcap_u8_t::value`

Value

Definition at line 914 of file types.h.

### 6.164.2.2 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](#)

## 6.165 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](#)

### 6.165.1 Detailed Description

VCAP UDP/TCP port range.

Definition at line 975 of file types.h.

### 6.165.2 Field Documentation

#### 6.165.2.1 in\_range

`BOOL vtss_vcap_udp_tcp_t::in_range`

Port in range match

Definition at line 977 of file types.h.

### 6.165.2.2 low

`vtss_udp_tcp_t vtss_vcap_udp_tcp_t::low`

Port low value

Definition at line 978 of file types.h.

### 6.165.2.3 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`

## 6.166 vtss\_vcap\_vid\_t Struct Reference

VCAP VLAN ID value and mask.

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

### Data Fields

- `u16 value`
- `u16 mask`

### 6.166.1 Detailed Description

VCAP VLAN ID value and mask.

Definition at line 961 of file types.h.

### 6.166.2 Field Documentation

### 6.166.2.1 value

`u16 vtss_vcap_vid_t::value`

Value

Definition at line 963 of file types.h.

### 6.166.2.2 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`

## 6.167 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`

### 6.167.1 Detailed Description

VCAP universal value or range.

Definition at line 994 of file types.h.

## 6.167.2 Field Documentation

### 6.167.2.1 type

`vtss_vcap_vr_type_t vtss_vcap_vr_t::type`

Type

Definition at line 996 of file types.h.

### 6.167.2.2 value

`vtss_vcap_vr_value_t vtss_vcap_vr_t::value`

Value

Definition at line 1001 of file types.h.

### 6.167.2.3 mask

`vtss_vcap_vr_value_t vtss_vcap_vr_t::mask`

Mask, cleared bits are wildcards

Definition at line 1002 of file types.h.

### 6.167.2.4 v

`struct { ... } vtss_vcap_vr_t::v`

`type == VTSS_VCAP_VR_TYPE_VALUE_MASK`

### 6.167.2.5 low

`vtss_vcap_vr_value_t vtss_vcap_vr_t::low`

Low value

Definition at line 1006 of file types.h.

### 6.167.2.6 high

```
vtss_vcap_vr_value_t vtss_vcap_vr_t::high
```

High value

Definition at line 1007 of file types.h.

### 6.167.2.7 r

```
struct { ... } vtss_vcap_vr_t::r  
type == VTSS_VCAP_VR_TYPE_RANGE_XXXXXX
```

### 6.167.2.8 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](#)

## 6.168 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](#)

### 6.168.1 Detailed Description

VCE Action.

Definition at line 1006 of file vtss\_l2\_api.h.

### 6.168.2 Field Documentation

### 6.168.2.1 vid

`vtss_vid_t` `vtss_vce_action_t::vid`

Classified VLAN ID

Definition at line 1008 of file `vtss_l2_api.h`.

### 6.168.2.2 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`

## 6.169 vtss\_vce\_frame\_etype\_t Struct Reference

Frame data for VTSS\_VCE\_TYPE\_ETYPE.

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

### Data Fields

- `vtss_vcap_u16_t` `etype`
- `vtss_vcap_u32_t` `data`

### 6.169.1 Detailed Description

Frame data for VTSS\_VCE\_TYPE\_ETYPE.

Definition at line 948 of file `vtss_l2_api.h`.

### 6.169.2 Field Documentation

### 6.169.2.1 etype

`vtss_vcap_u16_t vtss_vce_frame_etype_t::etype`

Ethernet Type value

Definition at line 950 of file `vtss_l2_api.h`.

### 6.169.2.2 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`

## 6.170 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`

### 6.170.1 Detailed Description

Frame data for VTSS\_VCE\_TYPE\_IPV4.

Definition at line 967 of file `vtss_l2_api.h`.

### 6.170.2 Field Documentation

### 6.170.2.1 fragment

`vtss_vcap_bit_t` vtss\_vce\_frame\_ipv4\_t::fragment

Fragment

Definition at line 969 of file vtss\_l2\_api.h.

### 6.170.2.2 options

`vtss_vcap_bit_t` vtss\_vce\_frame\_ipv4\_t::options

Header options

Definition at line 970 of file vtss\_l2\_api.h.

### 6.170.2.3 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.

### 6.170.2.4 proto

`vtss_vcap_u8_t` vtss\_vce\_frame\_ipv4\_t::proto

Protocol

Definition at line 972 of file vtss\_l2\_api.h.

### 6.170.2.5 sip

`vtss_vcap_ip_t` vtss\_vce\_frame\_ipv4\_t::sip

Source IP address

Definition at line 973 of file vtss\_l2\_api.h.

### 6.170.2.6 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`

## 6.171 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`

### 6.171.1 Detailed Description

Frame data for VTSS\_VCE\_TYPE\_IPV6.

Definition at line 978 of file vtss\_l2\_api.h.

### 6.171.2 Field Documentation

#### 6.171.2.1 dscp

`vtss_vcap_vr_t vtss_vce_frame_ipv6_t::dscp`

DSCP field (6 bit)

Definition at line 980 of file vtss\_l2\_api.h.

### 6.171.2.2 proto

`vtss_vcap_u8_t` `vtss_vce_frame_ipv6_t::proto`

Protocol

Definition at line 981 of file vtss\_l2\_api.h.

### 6.171.2.3 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.

### 6.171.2.4 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`

## 6.172 vtss\_vce\_frame\_llc\_t Struct Reference

Frame data for VTSS\_VCE\_TYPE LLC.

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

### Data Fields

- `vtss_vcap_u48_t data`

### 6.172.1 Detailed Description

Frame data for VTSS\_VCE\_TYPE LLC.

Definition at line 955 of file vtss\_l2\_api.h.

## 6.172.2 Field Documentation

### 6.172.2.1 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`

## 6.173 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`

### 6.173.1 Detailed Description

Frame data for VTSS\_VCE\_TYPE\_SNAP.

Definition at line 961 of file vtss\_l2\_api.h.

### 6.173.2 Field Documentation

#### 6.173.2.1 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`

## 6.174 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`

### 6.174.1 Detailed Description

VCE Key.

Definition at line 987 of file vtss\_l2\_api.h.

### 6.174.2 Field Documentation

#### 6.174.2.1 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.

#### 6.174.2.2 mac

```
vtss_vce_mac_t vtss_vce_key_t::mac
```

MAC header

Definition at line 990 of file vtss\_l2\_api.h.

#### 6.174.2.3 tag

`vtss_vce_tag_t` `vtss_vce_key_t::tag`

Tag

Definition at line 991 of file vtss\_l2\_api.h.

#### 6.174.2.4 type

`vtss_vce_type_t` `vtss_vce_key_t::type`

VCE frame type

Definition at line 992 of file vtss\_l2\_api.h.

#### 6.174.2.5 etype

`vtss_vce_frame_etype_t` `vtss_vce_key_t::etype`

VTSS\_VCE\_TYPE\_ETYPE

Definition at line 997 of file vtss\_l2\_api.h.

#### 6.174.2.6 llc

`vtss_vce_frame_llc_t` `vtss_vce_key_t::llc`

VTSS\_VCE\_TYPE\_LLCC

Definition at line 998 of file vtss\_l2\_api.h.

#### 6.174.2.7 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.

#### 6.174.2.8 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`.

#### 6.174.2.9 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`.

#### 6.174.2.10 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`

## 6.175 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`

#### 6.175.1 Detailed Description

VCE MAC header information.

Definition at line 930 of file `vtss_l2_api.h`.

## 6.175.2 Field Documentation

### 6.175.2.1 dmac\_mc

`vtss_vcap_bit_t` `vtss_vce_mac_t::dmac_mc`

Multicast DMAC

Definition at line 932 of file `vtss_l2_api.h`.

### 6.175.2.2 dmac\_bc

`vtss_vcap_bit_t` `vtss_vce_mac_t::dmac_bc`

Broadcast DMAC

Definition at line 933 of file `vtss_l2_api.h`.

### 6.175.2.3 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`

## 6.176 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`

### 6.176.1 Detailed Description

VLAN Control Entry.

Definition at line 1013 of file vtss\_l2\_api.h.

### 6.176.2 Field Documentation

#### 6.176.2.1 id

`vtss_vce_id_t` `vtss_vce_t::id`

VCE ID

Definition at line 1015 of file vtss\_l2\_api.h.

#### 6.176.2.2 key

`vtss_vce_key_t` `vtss_vce_t::key`

VCE Key

Definition at line 1016 of file vtss\_l2\_api.h.

#### 6.176.2.3 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](#)

## 6.177 **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`

### 6.177.1 Detailed Description

VCE tag information.

Definition at line 938 of file vtss\_l2\_api.h.

### 6.177.2 Field Documentation

#### 6.177.2.1 vid

`vtss_vcap_vid_t vtss_vce_tag_t::vid`

VLAN ID (12 bit)

Definition at line 940 of file vtss\_l2\_api.h.

#### 6.177.2.2 pcp

`vtss_vcap_u8_t vtss_vce_tag_t::pcp`

PCP (3 bit)

Definition at line 941 of file vtss\_l2\_api.h.

#### 6.177.2.3 dei

`vtss_vcap_bit_t vtss_vce_tag_t::dei`

DEI

Definition at line 942 of file vtss\_l2\_api.h.

#### 6.177.2.4 tagged

`vtss_vcap_bit_t` `vtss_vce_tag_t::tagged`

Tagged/untagged frame

Definition at line 943 of file `vtss_l2_api.h`.

#### 6.177.2.5 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`

## 6.178 vtss\_vcl\_port\_conf\_t Struct Reference

VCL port configuration.

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

### Data Fields

- `BOOL dmac_dip`

#### 6.178.1 Detailed Description

VCL port configuration.

Definition at line 878 of file `vtss_l2_api.h`.

#### 6.178.2 Field Documentation

### 6.178.2.1 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`

## 6.179 vtss\_vid\_mac\_t Struct Reference

MAC Address in specific VLAN.

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

### Data Fields

- `vtss_vid_t vid`
- `vtss_mac_t mac`

### 6.179.1 Detailed Description

MAC Address in specific VLAN.

Definition at line 652 of file `types.h`.

### 6.179.2 Field Documentation

#### 6.179.2.1 vid

`vtss_vid_t vtss_vid_mac_t::vid`

VLAN ID

Definition at line 654 of file `types.h`.

### 6.179.2.2 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](#)

## 6.180 vtss\_vlan\_conf\_t Struct Reference

VLAN configuration.

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

### Data Fields

- [vtss\\_etype\\_t s\\_etype](#)

### 6.180.1 Detailed Description

VLAN configuration.

Definition at line 625 of file vtss\_l2\_api.h.

### 6.180.2 Field Documentation

#### 6.180.2.1 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](#)

## 6.181 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`

#### 6.181.1 Detailed Description

VLAN port configuration.

Definition at line 664 of file `vtss_l2_api.h`.

#### 6.181.2 Field Documentation

##### 6.181.2.1 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`.

##### 6.181.2.2 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`.

### 6.181.2.3 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.

### 6.181.2.4 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.

### 6.181.2.5 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](#)

## 6.182 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`

### 6.182.1 Detailed Description

VLAN tag with "arbitrary" TPID.

Definition at line 632 of file types.h.

## 6.182.2 Field Documentation

### 6.182.2.1 tpid

`vtss_etype_t` `vtss_vlan_tag_t::tpid`

Tag Protocol Identifier

Definition at line 633 of file types.h.

### 6.182.2.2 pcp

`vtss_tagprio_t` `vtss_vlan_tag_t::pcp`

Priority Code Point

Definition at line 634 of file types.h.

### 6.182.2.3 dei

`BOOL` `vtss_vlan_tag_t::dei`

Drop Eligible Indicator

Definition at line 635 of file types.h.

### 6.182.2.4 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`

## 6.183 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`

#### 6.183.1 Detailed Description

VLAN translation group-to-VLAN configuration.

Definition at line 1104 of file `vtss_l2_api.h`.

#### 6.183.2 Field Documentation

##### 6.183.2.1 group\_id

```
u16 vtss_vlan_trans_grp2vlan_conf_t::group_id
```

Group ID

Definition at line 1105 of file `vtss_l2_api.h`.

##### 6.183.2.2 vid

```
vtss_vid_t vtss_vlan_trans_grp2vlan_conf_t::vid
```

VLAN ID

Definition at line 1106 of file `vtss_l2_api.h`.

### 6.183.2.3 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`

## 6.184 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]`

### 6.184.1 Detailed Description

VLAN translation port-to-group configuration.

Definition at line 1098 of file `vtss_l2_api.h`.

### 6.184.2 Field Documentation

#### 6.184.2.1 group\_id

`u16 vtss_vlan_trans_port2grp_conf_t::group_id`

Group ID

Definition at line 1099 of file `vtss_l2_api.h`.

### 6.184.2.2 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

## 6.185 vtss\_vlan\_vid\_conf\_t Struct Reference

VLAN ID configuration.

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

### Data Fields

- `BOOL learning`
- `BOOL mirror`

#### 6.185.1 Detailed Description

VLAN ID configuration.

Definition at line 740 of file vtss\_l2\_api.h.

#### 6.185.2 Field Documentation

##### 6.185.2.1 learning

```
BOOL vtss_vlan_vid_conf_t::learning
```

Enable/disable learning

Definition at line 742 of file vtss\_l2\_api.h.

### 6.185.2.2 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`

## 6.186 `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]`

### 6.186.1 Detailed Description

Structure for Wake-On-LAN MAC Address.

Definition at line 1656 of file `vtss_phy_api.h`.

### 6.186.2 Field Documentation

#### 6.186.2.1 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 7

## File Documentation

### 7.1 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.*

#### 7.1.1 Detailed Description

Layer 2 Public API Header for l2.

This header file describes public Layer 2 datatypes

#### 7.1.2 Enumeration Type Documentation

##### 7.1.2.1 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.

## 7.2 vtss\_api/include/vtss/api/options.h File Reference

Features and options.

### Macros

- #define VTSS\_ARCH\_CARACAL
- #define VTSS\_FEATURE\_EVC
- #define VTSS\_FEATURE\_QOS\_POLICER\_DLDB
- #define VTSS\_ARCH\_LUTON26
- #define VTSS\_FEATURE\_MISC
- #define VTSS\_FEATURE\_SERIAL\_GPIO
- #define VTSS\_FEATURE\_PORT\_CONTROL
- #define VTSS\_FEATURE\_CLAUSE\_37
- #define VTSS\_FEATURE\_EXC\_COL\_CONT
- #define VTSS\_FEATURE\_PORT\_CNT\_BRIDGE
- #define VTSS\_FEATURE\_QOS
- #define VTSS\_FEATURE\_QCL
- #define VTSS\_FEATURE\_QCL\_V2
- #define VTSS\_FEATURE\_QCL\_DMAC\_DIP
- #define VTSS\_FEATURE\_QCL\_KEY\_S\_TAG
- #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\_QUEUE\_POLICER
- #define VTSS\_FEATURE\_QOS\_QUEUE\_TX
- #define VTSS\_FEATURE\_QOS\_SCHEDULER\_V2
- #define VTSS\_FEATURE\_QOS\_TAG\_REMARK\_V2
- #define VTSS\_FEATURE\_QOS\_CLASSIFICATION\_V2
- #define VTSS\_FEATURE\_QOS\_EGRESS\_QUEUE\_SHAPERS
- #define VTSS\_FEATURE\_QOS\_EGRESS\_QUEUE\_SHAPERS\_EB
- #define VTSS\_FEATURE\_QOS\_DSCP\_CLASS\_DP\_AWARE
- #define VTSS\_FEATURE\_QOS\_DSCP\_REMARK
- #define VTSS\_FEATURE\_QOS\_DSCP\_REMARK\_V2
- #define VTSS\_FEATURE\_QOS\_DSCP\_REMARK\_DP\_AWARE
- #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\_IPV4\_MC\_SIP
- #define VTSS\_FEATURE\_IPV6\_MC\_SIP
- #define VTSS\_FEATURE\_MAC\_AGE\_AUTO
- #define VTSS\_FEATURE\_MAC\_CPU\_QUEUE
- #define VTSS\_FEATURE\_EEE
- #define VTSS\_FEATURE\_PORT\_MUX
- #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\_TIMESTAMP\_ONE\_STEP
- #define VTSS\_FEATURE\_TIMESTAMP\_LATENCY\_COMP
- #define VTSS\_FEATURE\_SYNC
- #define VTSS\_FEATURE\_NPI
- #define VTSS\_FEATURE\_IRQ\_CONTROL
- #define VTSS\_FEATURE\_LED\_POW\_REDUC
- #define VTSS\_FEATURE\_INTERRUPTS
- #define VTSS\_FEATURE\_VLAN\_TRANSLATION
- #define VTSS\_FEATURE\_SFLOW
- #define VTSS\_FEATURE\_MIRROR\_CPU
- #define VTSS\_FEATURE\_SERDES\_MACRO\_SETTINGS
- #define VTSS\_CHIP CU PHY
- #define VTSS\_OPT TRACE 1
- #define VTSS\_OPT\_VAUI\_EQ\_CTRL 6
- #define VTSS\_PHY\_OPT\_VERIPHYSY 1
- #define VTSS\_FEATURE\_WARM\_START
- #define VTSS\_FEATURE\_VCAP

### 7.2.1 Detailed Description

Features and options.

This header file describes target features and compile-time options

### 7.2.2 Macro Definition Documentation

### 7.2.2.1 VTSS\_ARCH\_CARACAL

```
#define VTSS_ARCH_CARACAL
```

Caracal CE switches

Definition at line 323 of file options.h.

### 7.2.2.2 VTSS\_FEATURE\_EVC

```
#define VTSS_FEATURE_EVC
```

Ethernet Virtual Connections

Definition at line 324 of file options.h.

### 7.2.2.3 VTSS\_FEATURE\_QOS\_POLICER\_DLB

```
#define VTSS_FEATURE_QOS_POLICER_DLB
```

DLB policers

Definition at line 328 of file options.h.

### 7.2.2.4 VTSS\_ARCH\_LUTON26

```
#define VTSS_ARCH_LUTON26
```

Luton26 architecture

Definition at line 332 of file options.h.

### 7.2.2.5 VTSS\_FEATURE\_MISC

```
#define VTSS_FEATURE_MISC
```

Miscellaneous

Definition at line 333 of file options.h.

### 7.2.2.6 VTSS\_FEATURE\_SERIAL\_GPIO

```
#define VTSS_FEATURE_SERIAL_GPIO
```

Serial GPIO control

Definition at line 334 of file options.h.

### 7.2.2.7 VTSS\_FEATURE\_PORT\_CONTROL

```
#define VTSS_FEATURE_PORT_CONTROL
```

Port control

Definition at line 335 of file options.h.

### 7.2.2.8 VTSS\_FEATURE\_CLAUSE\_37

```
#define VTSS_FEATURE_CLAUSE_37
```

IEEE 802.3 clause 37 auto-negotiation

Definition at line 336 of file options.h.

### 7.2.2.9 VTSS\_FEATURE\_EXC\_COL\_CONT

```
#define VTSS_FEATURE_EXC_COL_CONT
```

Excessive collision continuation

Definition at line 337 of file options.h.

### 7.2.2.10 VTSS\_FEATURE\_PORT\_CNT\_BRIDGE

```
#define VTSS_FEATURE_PORT_CNT_BRIDGE
```

Bridge counters

Definition at line 338 of file options.h.

### 7.2.2.11 VTSS\_FEATURE\_QOS

```
#define VTSS_FEATURE_QOS
```

QoS

Definition at line 339 of file options.h.

### 7.2.2.12 VTSS\_FEATURE\_QCL

```
#define VTSS_FEATURE_QCL
```

QoS: QoS Control Lists

Definition at line 340 of file options.h.

### 7.2.2.13 VTSS\_FEATURE\_QCL\_V2

```
#define VTSS_FEATURE_QCL_V2
```

QoS: QoS Control Lists, V2 features

Definition at line 341 of file options.h.

### 7.2.2.14 VTSS\_FEATURE\_QCL\_DMAC\_DIP

```
#define VTSS_FEATURE_QCL_DMAC_DIP
```

QoS: QoS Control Lists, match on either SMAC/SIP or DMAC/DIP

Definition at line 342 of file options.h.

### 7.2.2.15 VTSS\_FEATURE\_QCL\_KEY\_S\_TAG

```
#define VTSS_FEATURE_QCL_KEY_S_TAG
```

QoS: QoS Control Lists has S tag support

Definition at line 343 of file options.h.

### 7.2.2.16 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 344 of file options.h.

### 7.2.2.17 VTSS\_FEATURE\_QCL\_POLICY\_ACTION

```
#define VTSS_FEATURE_QCL_POLICY_ACTION
```

QoS: QoS Control Lists has policy action

Definition at line 345 of file options.h.

### 7.2.2.18 VTSS\_FEATURE\_QOS\_POLICER\_UC\_SWITCH

```
#define VTSS_FEATURE_QOS_POLICER_UC_SWITCH
```

QoS: Unicast policer per switch

Definition at line 346 of file options.h.

### 7.2.2.19 VTSS\_FEATURE\_QOS\_POLICER\_MC\_SWITCH

```
#define VTSS_FEATURE_QOS_POLICER_MC_SWITCH
```

QoS: Multicast policer per switch

Definition at line 347 of file options.h.

### 7.2.2.20 VTSS\_FEATURE\_QOS\_POLICER\_BC\_SWITCH

```
#define VTSS_FEATURE_QOS_POLICER_BC_SWITCH
```

QoS: Broadcast policer per switch

Definition at line 348 of file options.h.

### 7.2.2.21 VTSS FEATURE QOS PORT POLICER EXT

```
#define VTSS_FEATURE_QOS_PORT_POLICER_EXT
```

QoS: Port Policer Extensions

Definition at line 349 of file options.h.

### 7.2.2.22 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 350 of file options.h.

### 7.2.2.23 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 351 of file options.h.

### 7.2.2.24 VTSS FEATURE QOS QUEUE POLICER

```
#define VTSS_FEATURE_QOS_QUEUE_POLICER
```

QoS: Has Ingress Queue Policers

Definition at line 352 of file options.h.

### 7.2.2.25 VTSS FEATURE QOS QUEUE TX

```
#define VTSS_FEATURE_QOS_QUEUE_TX
```

QoS: Has TX Queue support

Definition at line 353 of file options.h.

**7.2.2.26 VTSS\_FEATURE\_QOS\_SCHEDULER\_V2**

```
#define VTSS_FEATURE_QOS_SCHEDULER_V2
```

QoS: 2. version of scheduler

Definition at line 354 of file options.h.

**7.2.2.27 VTSS\_FEATURE\_QOS\_TAG\_REMARK\_V2**

```
#define VTSS_FEATURE_QOS_TAG_REMARK_V2
```

QoS: 2. version of tag priority remarking

Definition at line 355 of file options.h.

**7.2.2.28 VTSS\_FEATURE\_QOS\_CLASSIFICATION\_V2**

```
#define VTSS_FEATURE_QOS_CLASSIFICATION_V2
```

QoS: 2. version of classification

Definition at line 356 of file options.h.

**7.2.2.29 VTSS\_FEATURE\_QOS\_EGRESS\_QUEUE\_SHAPERS**

```
#define VTSS_FEATURE_QOS_EGRESS_QUEUE_SHAPERS
```

QoS: Has Egress Queue Shapers

Definition at line 357 of file options.h.

**7.2.2.30 VTSS\_FEATURE\_QOS\_EGRESS\_QUEUE\_SHAPERS\_EB**

```
#define VTSS_FEATURE_QOS_EGRESS_QUEUE_SHAPERS_EB
```

QoS: Egress Queue Shapers has Excess Bandwidth support

Definition at line 358 of file options.h.

### 7.2.2.31 VTSS FEATURE QOS\_DSCP\_CLASS\_DP\_AWARE

```
#define VTSS_FEATURE_QOS_DSCP_CLASS_DP_AWARE
```

QoS: DSCP classification is DP aware

Definition at line 359 of file options.h.

### 7.2.2.32 VTSS FEATURE QOS\_DSCP\_REMARK

```
#define VTSS_FEATURE_QOS_DSCP_REMARK
```

QoS: Has DSCP remarking

Definition at line 360 of file options.h.

### 7.2.2.33 VTSS FEATURE QOS\_DSCP\_REMARK\_V2

```
#define VTSS_FEATURE_QOS_DSCP_REMARK_V2
```

QoS: 2. version of DSCP remarking

Definition at line 361 of file options.h.

### 7.2.2.34 VTSS FEATURE QOS\_DSCP\_REMARK\_DP\_AWARE

```
#define VTSS_FEATURE_QOS_DSCP_REMARK_DP_AWARE
```

QoS: DSCP remarking is DP aware

Definition at line 362 of file options.h.

### 7.2.2.35 VTSS FEATURE PACKET

```
#define VTSS_FEATURE_PACKET
```

CPU Rx/Tx frame configuration

Definition at line 363 of file options.h.

### 7.2.2.36 VTSS FEATURE PACKET TX

```
#define VTSS_FEATURE_PACKET_TX
```

CPU Tx frame

Definition at line 364 of file options.h.

### 7.2.2.37 VTSS FEATURE PACKET RX

```
#define VTSS_FEATURE_PACKET_RX
```

CPU Rx frame

Definition at line 365 of file options.h.

### 7.2.2.38 VTSS FEATURE PACKET GROUPING

```
#define VTSS_FEATURE_PACKET_GROUPING
```

Extraction and injection occurs through extraction and injection groups rather than queues.

Definition at line 366 of file options.h.

### 7.2.2.39 VTSS FEATURE PACKET PORT REG

```
#define VTSS_FEATURE_PACKET_PORT_REG
```

Packet registration per port

Definition at line 367 of file options.h.

### 7.2.2.40 VTSS FEATURE LAYER2

```
#define VTSS_FEATURE_LAYER2
```

Layer 2 (switching)

Definition at line 368 of file options.h.

#### 7.2.2.41 VTSS\_FEATURE\_PVLAN

```
#define VTSS_FEATURE_PVLAN
```

Private VLANs

Definition at line 369 of file options.h.

#### 7.2.2.42 VTSS\_FEATURE\_VLAN\_PORT\_V2

```
#define VTSS_FEATURE_VLAN_PORT_V2
```

VLAN port configuration, V2 features

Definition at line 370 of file options.h.

#### 7.2.2.43 VTSS\_FEATURE\_VLAN\_TX\_TAG

```
#define VTSS_FEATURE_VLAN_TX_TAG
```

VLAN tagging per (VID, port)

Definition at line 371 of file options.h.

#### 7.2.2.44 VTSS\_FEATURE\_IPV4\_MC\_SIP

```
#define VTSS_FEATURE_IPV4_MC_SIP
```

Source specific IPv4 multicast

Definition at line 372 of file options.h.

#### 7.2.2.45 VTSS\_FEATURE\_IPV6\_MC\_SIP

```
#define VTSS_FEATURE_IPV6_MC_SIP
```

Source specific IPv6 multicast

Definition at line 373 of file options.h.

### 7.2.2.46 VTSS\_FEATURE\_MAC\_AGE\_AUTO

```
#define VTSS_FEATURE_MAC_AGE_AUTO
```

Automatic MAC address ageing

Definition at line 374 of file options.h.

### 7.2.2.47 VTSS\_FEATURE\_MAC\_CPU\_QUEUE

```
#define VTSS_FEATURE_MAC_CPU_QUEUE
```

CPU queue per MAC address

Definition at line 375 of file options.h.

### 7.2.2.48 VTSS\_FEATURE\_EEE

```
#define VTSS_FEATURE_EEE
```

Energy Efficient Ethernet

Definition at line 376 of file options.h.

### 7.2.2.49 VTSS\_FEATURE\_PORT\_MUX

```
#define VTSS_FEATURE_PORT_MUX
```

Port mux between serdes blocks and ports

Definition at line 377 of file options.h.

### 7.2.2.50 VTSS\_FEATURE\_FAN

```
#define VTSS_FEATURE_FAN
```

Fan control

Definition at line 378 of file options.h.

**7.2.2.51 VTSS\_FEATURE\_VCAP [1/2]**

```
#define VTSS_FEATURE_VCAP
```

VCAP

Definition at line 637 of file options.h.

**7.2.2.52 VTSS\_FEATURE\_ACL**

```
#define VTSS_FEATURE_ACL
```

Access Control Lists

Definition at line 380 of file options.h.

**7.2.2.53 VTSS\_FEATURE\_ACL\_V2**

```
#define VTSS_FEATURE_ACL_V2
```

Access Control Lists, V2 features

Definition at line 381 of file options.h.

**7.2.2.54 VTSS\_FEATURE\_VCL**

```
#define VTSS_FEATURE_VCL
```

VLAN Control Lists

Definition at line 382 of file options.h.

**7.2.2.55 VTSS\_FEATURE\_TIMESTAMP**

```
#define VTSS_FEATURE_TIMESTAMP
```

Packet timestamp feature (for PTP and OAM)

Definition at line 383 of file options.h.

**7.2.2.56 VTSS\_FEATURE\_TIMESTAMP\_ONE\_STEP**

```
#define VTSS_FEATURE_TIMESTAMP_ONE_STEP
```

ONESTEP timestamp hardware support

Definition at line 384 of file options.h.

**7.2.2.57 VTSS\_FEATURE\_TIMESTAMP\_LATENCY\_COMP**

```
#define VTSS_FEATURE_TIMESTAMP_LATENCY_COMP
```

Ingress and egress latency compensation hardware support

Definition at line 385 of file options.h.

**7.2.2.58 VTSS\_FEATURE\_SYNCE**

```
#define VTSS_FEATURE_SYNCE
```

SYNCE - L1 synchronization feature

Definition at line 388 of file options.h.

**7.2.2.59 VTSS\_FEATURE\_NPI**

```
#define VTSS_FEATURE_NPI
```

NPI port

Definition at line 389 of file options.h.

**7.2.2.60 VTSS\_FEATURE\_IRQ\_CONTROL**

```
#define VTSS_FEATURE_IRQ_CONTROL
```

General IRQ support

Definition at line 390 of file options.h.

### 7.2.2.61 VTSS\_FEATURE\_LED\_POW\_REDUC

```
#define VTSS_FEATURE_LED_POW_REDUC
```

LED power reduction

Definition at line 400 of file options.h.

### 7.2.2.62 VTSS\_FEATURE\_INTERRUPTS

```
#define VTSS_FEATURE_INTERRUPTS
```

Port Interrupt support

Definition at line 401 of file options.h.

### 7.2.2.63 VTSS\_FEATURE\_VLAN\_TRANSLATION

```
#define VTSS_FEATURE_VLAN_TRANSLATION
```

VLAN Translation

Definition at line 402 of file options.h.

### 7.2.2.64 VTSS\_FEATURE\_SFLOW

```
#define VTSS_FEATURE_SFLOW
```

Statistical flow sampling

Definition at line 403 of file options.h.

### 7.2.2.65 VTSS\_FEATURE\_MIRROR\_CPU

```
#define VTSS_FEATURE_MIRROR_CPU
```

CPU mirroring

Definition at line 404 of file options.h.

### 7.2.2.66 VTSS\_FEATURE\_SERDES\_MACRO\_SETTINGS

```
#define VTSS_FEATURE_SERDES_MACRO_SETTINGS
```

Hooks for Serdes Macro configuration

Definition at line 405 of file options.h.

### 7.2.2.67 VTSS\_CHIP CU PHY

```
#define VTSS_CHIP CU PHY
```

Copper PHY chip

Definition at line 540 of file options.h.

### 7.2.2.68 VTSS\_OPT\_TRACE

```
#define VTSS_OPT_TRACE 1
```

Trace enabled by default

Definition at line 548 of file options.h.

### 7.2.2.69 VTSS\_OPT\_VAUI\_EQ\_CTRL

```
#define VTSS_OPT_VAUI_EQ_CTRL 6
```

Default equalization control

Definition at line 605 of file options.h.

### 7.2.2.70 VTSS\_PHY\_OPT\_VERIPHY

```
#define VTSS_PHY_OPT_VERIPHY 1
```

VeriPHY enabled by default

Definition at line 613 of file options.h.

### 7.2.2.71 VTSS\_FEATURE\_WARM\_START

```
#define VTSS_FEATURE_WARM_START
```

Warm start

Definition at line 617 of file options.h.

### 7.2.2.72 VTSS\_FEATURE\_VCAP [2/2]

```
#define VTSS_FEATURE_VCAP
```

VCAP

Definition at line 637 of file options.h.

## 7.3 vtss\_api/include/vtss/api/phy.h File Reference

PHY Public API Header.

```
#include <vtss/api/types.h>
```

### Macros

- `#define VTSS_PHY_POWER_ACTIPHYS_BIT 0`
- `#define VTSS_PHY_POWER_DYNAMIC_BIT 1`

### Enumerations

- enum `vtss_phy_power_mode_t` { `VTSS_PHY_POWER_NOMINAL` = 0, `VTSS_PHY_POWER_ACTIPHYS` = 1 << `VTSS_PHY_POWER_ACTIPHYS_BIT`, `VTSS_PHY_POWER_DYNAMIC` = 1 << `VTSS_PHY_POWER_DYNAMIC_BIT`, `VTSS_PHY_POWER_ENABLED` = `VTSS_PHY_POWER_ACTIPHYS` + `VTSS_PHY_POWER_DYNAMIC` }
- PHY power reduction modes.*
- enum `vtss_phy_veriphy_status_t` { `VTSS_VERIPHYS_STATUS_OK` = 0, `VTSS_VERIPHYS_STATUS_OPEN` = 1, `VTSS_VERIPHYS_STATUS_SHORT` = 2, `VTSS_VERIPHYS_STATUS_ABNORM` = 4, `VTSS_VERIPHYS_STATUS_SHORT_A` = 8, `VTSS_VERIPHYS_STATUS_SHORT_B` = 9, `VTSS_VERIPHYS_STATUS_SHORT_C` = 10, `VTSS_VERIPHYS_STATUS_SHORT_D` = 11, `VTSS_VERIPHYS_STATUS_COUPL_A` = 12, `VTSS_VERIPHYS_STATUS_COUPL_B` = 13, `VTSS_VERIPHYS_STATUS_COUPL_C` = 14, `VTSS_VERIPHYS_STATUS_COUPL_D` = 15, `VTSS_VERIPHYS_STATUS_UNKNOWN` = 16, `VTSS_VERIPHYS_STATUS_RUNNING` = 17 }

*VeriPHY status.*

### 7.3.1 Detailed Description

PHY Public API Header.

This header file describes public PHY data-types

### 7.3.2 Macro Definition Documentation

#### 7.3.2.1 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.

#### 7.3.2.2 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.

### 7.3.3 Enumeration Type Documentation

#### 7.3.3.1 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.

### 7.3.3.2 vtss\_phy\_veriphy\_status\_t

enum `vtss_phy_veriphy_status_t`

VeriPHY status.

Enumerator

<code>VTSS_VERIPHY_STATUS_OK</code>	Correctly terminated pair
<code>VTSS_VERIPHY_STATUS_OPEN</code>	Open pair
<code>VTSS_VERIPHY_STATUS_SHORT</code>	Short pair
<code>VTSS_VERIPHY_STATUS_ABNORM</code>	Abnormal termination
<code>VTSS_VERIPHY_STATUS_SHORT_A</code>	Cross-pair short to pair A
<code>VTSS_VERIPHY_STATUS_SHORT_B</code>	Cross-pair short to pair B
<code>VTSS_VERIPHY_STATUS_SHORT_C</code>	Cross-pair short to pair C
<code>VTSS_VERIPHY_STATUS_SHORT_D</code>	Cross-pair short to pair D
<code>VTSS_VERIPHY_STATUS_COUPL_A</code>	Abnormal cross-pair coupling, pair A
<code>VTSS_VERIPHY_STATUS_COUPL_B</code>	Abnormal cross-pair coupling, pair B
<code>VTSS_VERIPHY_STATUS_COUPL_C</code>	Abnormal cross-pair coupling, pair C
<code>VTSS_VERIPHY_STATUS_COUPL_D</code>	Abnormal cross-pair coupling, pair D
<code>VTSS_VERIPHY_STATUS_UNKNOWN</code>	Unknown - VeriPhy never started ?
<code>VTSS_VERIPHY_STATUS_RUNNING</code>	VeriPhy is still running - No result yet

Definition at line 54 of file phy.h.

## 7.4 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_evc_counters_t`  
*EVC counters.*
- struct `vtss_port_bridge_counters_t`  
*Port bridge counter structure (RFC 4188)*

- struct `vtss_port_proprietary_counters_t`  
*Port proprietary counter structure.*
- struct `vtss_port_counters_t`  
*Port counter structure.*
- struct `port_custom_conf_t`  
*Port configuration.*
- struct `vtss_port_status_t`  
*Port status parameter struct.*

## Macros

- `#define PORT_CAP_NONE 0x00000000`
- `#define PORT_CAP_AUTONEG 0x00000001`
- `#define PORT_CAP_10M_HDX 0x00000002`
- `#define PORT_CAP_10M_FDX 0x00000004`
- `#define PORT_CAP_100M_HDX 0x00000008`
- `#define PORT_CAP_100M_FDX 0x00000010`
- `#define PORT_CAP_1G_FDX 0x00000020`
- `#define PORT_CAP_2_5G_FDX 0x00000040`
- `#define PORT_CAP_5G_FDX 0x00000080`
- `#define PORT_CAP_10G_FDX 0x00000100`
- `#define PORT_CAP_FLOW_CTRL 0x00001000`
- `#define PORT_CAP_COPPER 0x00002000`
- `#define PORT_CAP_FIBER 0x00004000`
- `#define PORT_CAP_DUAL_COPPER 0x00008000`
- `#define PORT_CAP_DUAL_FIBER 0x00010000`
- `#define PORT_CAP_SD_ENABLE 0x00020000`
- `#define PORT_CAP_SD_HIGH 0x00040000`
- `#define PORT_CAP_SD_INTERNAL 0x00080000`
- `#define PORT_CAP_DUAL_FIBER_100FX 0x00100000`
- `#define PORT_CAP_XAUI_LANE_FLIP 0x00200000`
- `#define PORT_CAP_VTSS_10G_PHY 0x00400000`
- `#define PORT_CAP_SFP_DETECT 0x00800000`
- `#define PORT_CAP_STACKING 0x01000000`
- `#define PORT_CAP_DUAL_SFP_DETECT 0x02000000`
- `#define PORT_CAP_SFP_ONLY 0x04000000`
- `#define PORT_CAP_DUAL_COPPER_100FX 0x08000000`
- `#define PORT_CAP_HDX (PORT_CAP_10M_HDX | PORT_CAP_100M_HDX)`
- `#define PORT_CAP_TRI_SPEED_FDX (PORT_CAP_AUTONEG | PORT_CAP_1G_FDX | PORT_CAP_100M_FDX | PORT_CAP_10M_FDX | PORT_CAP_FLOW_CTRL)`
- `#define PORT_CAP_TRI_SPEED (PORT_CAP_TRI_SPEED_FDX | PORT_CAP_HDX)`
- `#define PORT_CAP_1G_PHY (PORT_CAP_COPPER | PORT_CAP_FIBER | PORT_CAP_DUAL_COPPER | PORT_CAP_DUAL_FIBER | PORT_CAP_DUAL_FIBER_100FX)`
- `#define PORT_CAP_TRI_SPEED_COPPER (PORT_CAP_TRI_SPEED | PORT_CAP_COPPER)`
- `#define PORT_CAP_TRI_SPEED_FIBER (PORT_CAP_TRI_SPEED | PORT_CAP_FIBER)`
- `#define PORT_CAP_TRI_SPEED_DUAL_COPPER (PORT_CAP_TRI_SPEED | PORT_CAP_DUAL_COPPER)`
- `#define PORT_CAP_TRI_SPEED_DUAL_FIBER (PORT_CAP_TRI_SPEED | PORT_CAP_DUAL_FIBER)`
- `#define PORT_CAP_TRI_SPEED_DUAL_FIBER_100FX (PORT_CAP_TRI_SPEED | PORT_CAP_DUAL_FIBER_100FX)`
- `#define PORT_CAP_ANY_FIBER (PORT_CAP_FIBER | PORT_CAP_DUAL_FIBER_100FX | PORT_CAP_DUAL_FIBER | PORT_CAP_DUAL_COPPER | PORT_CAP_SFP_DETECT)`
- `#define PORT_CAP_SPEED_DUAL_ANY_FIBER_FIXED_SPEED (PORT_CAP_DUAL_FIBER_100FX | PORT_CAP_DUAL_FIBER | PORT_CAP_DUAL_COPPER)`

- #define PORT\_CAP\_SPEED\_DUAL\_ANY\_FIBER (PORT\_CAP\_DUAL\_COPPER | PORT\_CAP\_DUAL\_FIBER | PORT\_CAP\_DUAL\_FIBER\_100FX | PORT\_CAP\_DUAL\_SFP\_DETECT)
- #define PORT\_CAP\_TRI\_SPEED\_DUAL\_ANY\_FIBER (PORT\_CAP\_TRI\_SPEED | PORT\_CAP\_SPEED\_DUAL\_ANY\_FIBER)
- #define PORT\_CAP\_TRI\_SPEED\_DUAL\_ANY\_FIBER\_FIXED\_SFP\_SPEED (PORT\_CAP\_TRI\_SPEED | PORT\_CAP\_SPEED\_DUAL\_ANY\_FIBER\_FIXED\_SPEED)
- #define PORT\_CAP\_DUAL\_FIBER\_1000X (PORT\_CAP\_DUAL\_FIBER | PORT\_CAP\_DUAL\_COPPER)
- #define PORT\_CAP\_SFP\_1G (PORT\_CAP\_AUTONEG | PORT\_CAP\_100M\_FDX | PORT\_CAP\_1G\_FDX | PORT\_CAP\_FLOW\_CTRL | PORT\_CAP\_SFP\_ONLY)
- #define PORT\_CAP\_SFP\_2\_5G (PORT\_CAP\_SFP\_1G | PORT\_CAP\_2\_5G\_FDX)
- #define PORT\_CAP\_SFP\_SD\_HIGH (PORT\_CAP\_SD\_ENABLE | PORT\_CAP\_SD\_HIGH | PORT\_CAP\_SD\_INTERNAL | PORT\_CAP\_SFP\_DETECT | PORT\_CAP\_SFP\_ONLY)
- #define PORT\_CAP\_2\_5G\_TRI\_SPEED\_FDX (PORT\_CAP\_AUTONEG | PORT\_CAP\_2\_5G\_FDX | PORT\_CAP\_1G\_FDX | PORT\_CAP\_100M\_FDX | PORT\_CAP\_FLOW\_CTRL)
- #define PORT\_CAP\_2\_5G\_TRI\_SPEED (PORT\_CAP\_2\_5G\_TRI\_SPEED\_FDX | PORT\_CAP\_100M\_HDX)
- #define PORT\_CAP\_2\_5G\_TRI\_SPEED\_COPPER (PORT\_CAP\_2\_5G\_TRI\_SPEED | PORT\_CAP\_COPPER)

## Typedefs

- typedef u32 port\_cap\_t
- typedef u64 vtss\_port\_counter\_t

*Counter type.*

## Enumerations

- enum vtss\_port\_speed\_t {
 VTSS\_SPEED\_UNDEFINED, VTSS\_SPEED\_10M, VTSS\_SPEED\_100M, VTSS\_SPEED\_1G,
 VTSS\_SPEED\_2500M, VTSS\_SPEED\_5G, VTSS\_SPEED\_10G, VTSS\_SPEED\_12G
 }
 

*Port speed.*
- enum vtss\_fiber\_port\_speed\_t {
 VTSS\_SPEED\_FIBER\_NOT\_SUPPORTED\_OR\_DISABLED = 0, VTSS\_SPEED\_FIBER\_100FX = 2,
 VTSS\_SPEED\_FIBER\_1000X = 3, VTSS\_SPEED\_FIBER\_AUTO = 4,
 VTSS\_SPEED\_FIBER\_DISABLED = 5
 }
 

*Fiber Port speed.*

### 7.4.1 Detailed Description

Port Public API Header.

This header file describes public port data-types

### 7.4.2 Macro Definition Documentation

#### 7.4.2.1 PORT\_CAP\_NONE

```
#define PORT_CAP_NONE 0x00000000
```

No capabilities

Definition at line 46 of file port.h.

#### 7.4.2.2 PORT\_CAP\_AUTONEG

```
#define PORT_CAP_AUTONEG 0x00000001
```

Auto negotiation

Definition at line 47 of file port.h.

#### 7.4.2.3 PORT\_CAP\_10M\_HDX

```
#define PORT_CAP_10M_HDX 0x00000002
```

10 Mbps, half duplex

Definition at line 48 of file port.h.

#### 7.4.2.4 PORT\_CAP\_10M\_FDX

```
#define PORT_CAP_10M_FDX 0x00000004
```

10 Mbps, full duplex

Definition at line 49 of file port.h.

#### 7.4.2.5 PORT\_CAP\_100M\_HDX

```
#define PORT_CAP_100M_HDX 0x00000008
```

100 Mbps, half duplex

Definition at line 50 of file port.h.

#### 7.4.2.6 PORT\_CAP\_100M\_FDX

```
#define PORT_CAP_100M_FDX 0x00000010
```

100 Mbps, full duplex

Definition at line 51 of file port.h.

#### 7.4.2.7 PORT\_CAP\_1G\_FDX

```
#define PORT_CAP_1G_FDX 0x00000020
```

1 Gbps, full duplex

Definition at line 52 of file port.h.

#### 7.4.2.8 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.

#### 7.4.2.9 PORT\_CAP\_5G\_FDX

```
#define PORT_CAP_5G_FDX 0x00000080
```

5Gbps, full duplex

Definition at line 54 of file port.h.

#### 7.4.2.10 PORT\_CAP\_10G\_FDX

```
#define PORT_CAP_10G_FDX 0x00000100
```

10Gbps, full duplex

Definition at line 55 of file port.h.

#### 7.4.2.11 PORT\_CAP\_FLOW\_CTRL

```
#define PORT_CAP_FLOW_CTRL 0x00001000
```

Flow control

Definition at line 56 of file port.h.

#### 7.4.2.12 PORT\_CAP\_COPPER

```
#define PORT_CAP_COPPER 0x00002000
```

Copper media

Definition at line 57 of file port.h.

#### 7.4.2.13 PORT\_CAP\_FIBER

```
#define PORT_CAP_FIBER 0x00004000
```

Fiber media

Definition at line 58 of file port.h.

#### 7.4.2.14 PORT\_CAP\_DUAL\_COPPER

```
#define PORT_CAP_DUAL_COPPER 0x00008000
```

Dual media, copper preferred

Definition at line 59 of file port.h.

#### 7.4.2.15 PORT\_CAP\_DUAL\_FIBER

```
#define PORT_CAP_DUAL_FIBER 0x00010000
```

Dual media, fiber preferred

Definition at line 60 of file port.h.

#### 7.4.2.16 PORT\_CAP\_SD\_ENABLE

```
#define PORT_CAP_SD_ENABLE 0x00020000
```

Signal Detect enabled

Definition at line 61 of file port.h.

#### 7.4.2.17 PORT\_CAP\_SD\_HIGH

```
#define PORT_CAP_SD_HIGH 0x00040000
```

Signal Detect active high

Definition at line 62 of file port.h.

#### 7.4.2.18 PORT\_CAP\_SD\_INTERNAL

```
#define PORT_CAP_SD_INTERNAL 0x00080000
```

Signal Detect select internal

Definition at line 63 of file port.h.

#### 7.4.2.19 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.

#### 7.4.2.20 PORT\_CAP\_XAUI\_LANE\_FLIP

```
#define PORT_CAP_XAUI_LANE_FLIP 0x00200000
```

Flip the XAUI lanes

Definition at line 65 of file port.h.

#### 7.4.2.21 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.

#### 7.4.2.22 PORT\_CAP\_SFP\_DETECT

```
#define PORT_CAP_SFP_DETECT 0x00800000
```

Auto detect the SFP module

Definition at line 67 of file port.h.

#### 7.4.2.23 PORT\_CAP\_STACKING

```
#define PORT_CAP_STACKING 0x01000000
```

Stack port candidate

Definition at line 68 of file port.h.

#### 7.4.2.24 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.

#### 7.4.2.25 PORT\_CAP\_SFP\_ONLY

```
#define PORT_CAP_SFP_ONLY 0x04000000
```

SFP only port (not dual media)

Definition at line 70 of file port.h.

#### 7.4.2.26 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.

#### 7.4.2.27 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.

#### 7.4.2.28 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.

#### 7.4.2.29 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.

#### 7.4.2.30 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.

#### 7.4.2.31 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.

#### 7.4.2.32 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.

#### 7.4.2.33 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.

#### 7.4.2.34 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.

#### 7.4.2.35 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.

#### 7.4.2.36 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.

#### 7.4.2.37 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.

#### 7.4.2.38 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.

#### 7.4.2.39 PORT\_CAP\_TRI\_SPEED\_DUAL\_ANY\_FIBER

```
#define PORT_CAP_TRI_SPEED_DUAL_ANY_FIBER (PORT_CAP_TRI_SPEED | PORT_CAP_SPEED_DUAL_ANY_FIBER)
```

Copper & Fiber mode, auto detection supported

Definition at line 87 of file port.h.

#### 7.4.2.40 PORT\_CAP\_TRI\_SPEED\_DUAL\_ANY\_FIBER\_FIXED\_SFP\_SPEED

```
#define PORT_CAP_TRI_SPEED_DUAL_ANY_FIBER_FIXED_SFP_SPEED (PORT_CAP_TRI_SPEED | PORT_CAP_SPEED_DUAL_ANY_FIBER_
```

Copper & Fiber mode, but SFP auto detection not supported

Definition at line 88 of file port.h.

#### 7.4.2.41 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.

#### 7.4.2.42 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.

#### 7.4.2.43 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.

#### 7.4.2.44 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.

#### 7.4.2.45 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.

#### 7.4.2.46 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.

#### 7.4.2.47 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.

### 7.4.3 Typedef Documentation

#### 7.4.3.1 port\_cap\_t

```
typedef u32 port_cap_t
```

Bit-mask containing the port capabilities

Definition at line 99 of file port.h.

### 7.4.4 Enumeration Type Documentation

#### 7.4.4.1 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.

#### 7.4.4.2 vtss\_fiber\_port\_speed\_t

enum [vtss\\_fiber\\_port\\_speed\\_t](#)

Fiber Port speed.

Enumerator

VTSS_SPEED_FIBER_NOT_SUPPORTED_OR_DISABLED	Fiber not supported/ Fiber port disabled
VTSS_SPEED_FIBER_100FX	100BASE-FX
VTSS_SPEED_FIBER_1000X	1000BASE-X
VTSS_SPEED_FIBER_AUTO	Auto detection
VTSS_SPEED_FIBER_DISABLED	Obsolete - use VTSS_SPEED_FIBER_NOT_SUPPORTED_OR_DISABLED instead

Definition at line 255 of file port.h.

## 7.5 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](#)

- struct [vtss\\_ipv6\\_uc\\_t](#)  
*IPv6 routing entry.*
- struct [vtss\\_routing\\_entry\\_t](#)  
*Routing entry.*
- struct [vtss\\_l3\\_counters\\_t](#)  
*Routing interface statics counter.*
- struct [vtss\\_vcap\\_u8\\_t](#)  
*VCAP 8 bit value and mask.*
- struct [vtss\\_vcap\\_u16\\_t](#)  
*VCAP 16 bit value and mask.*
- struct [vtss\\_vcap\\_u24\\_t](#)  
*VCAP 24 bit value and mask.*
- struct [vtss\\_vcap\\_u32\\_t](#)  
*VCAP 32 bit value and mask.*
- struct [vtss\\_vcap\\_u40\\_t](#)  
*VCAP 40 bit value and mask.*
- struct [vtss\\_vcap\\_u48\\_t](#)  
*VCAP 48 bit value and mask.*
- struct [vtss\\_vcap\\_u128\\_t](#)  
*VCAP 128 bit value and mask.*
- struct [vtss\\_vcap\\_vid\\_t](#)  
*VCAP VLAN ID value and mask.*
- struct [vtss\\_vcap\\_ip\\_t](#)  
*VCAP IPv4 address value and mask.*
- struct [vtss\\_vcap\\_udp\\_tcp\\_t](#)  
*VCAP UDP/TCP port range.*
- struct [vtss\\_vcap\\_vr\\_t](#)  
*VCAP universal value or range.*
- struct [vtss\\_counter\\_pair\\_t](#)  
*Counter pair.*
- struct [vtss\\_timestamp\\_t](#)  
*Time stamp in seconds and nanoseconds.*

## Macros

- #define PRId64 "lld"
- #define PRId64 "ld"
- #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 26
- #define VTSS\_PORTS VTSS\_OPT\_PORT\_COUNT
- #define VTSS\_PORT\_NO\_NONE (0xffffffff)

- #define VTSS\_PORT\_NO\_CPU (0xffffffff)
- #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\_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 0x7fffffffffffffLL`
- `#define VTSS_INTERVAL_SEC(t) ((i32)VTSS_DIV64((t)>>16, VTSS_ONE_MIA))`
- `#define VTSS_INTERVAL_MS(t) ((i32)VTSS_DIV64((t)>>16, VTSS_ONE_MILL))`
- `#define VTSS_INTERVAL_US(t) ((i32)VTSS_DIV64((t)>>16, 1000))`
- `#define VTSS_INTERVAL_NS(t) ((i32)VTSS_MOD64((t)>>16, VTSS_ONE_MIA))`
- `#define VTSS_INTERVAL_PS(t) (((i32)(t & 0xffff)*1000)+0x8000)/0x10000`
- `#define VTSS_SEC_NS_INTERVAL(s, n) (((vtss_timeinterval_t)(n)+(vtss_timeinterval_t)(s)*VTSS_ONE_MIA)<<16)`
- `#define VTSS_CLOCK_IDENTITY_LENGTH 8`
- `#define VTSS_SYNC_CLK_PORT_ARRAY_SIZE 2`

## Typedefs

- `typedef char i8`

*Fallback Integer types.*
- `typedef signed short i16`
- `typedef signed int i32`
- `typedef signed long long i64`
- `typedef unsigned char u8`
- `typedef unsigned short u16`
- `typedef unsigned int u32`
- `typedef unsigned long long u64`
- `typedef unsigned char BOOL`
- `typedef unsigned int uintptr_t`
- `typedef int vtss_rc`

*Error code type.*
- `typedef u32 vtss_chip_no_t`

*Chip number used for targets with multiple chips.*
- `typedef struct vtss_state_s * vtss_inst_t`

*Instance identifier.*
- `typedef BOOL vtss_event_t`

*Description: Event type. When a variable of this type is used as an input parameter, the API will set the variable if the event has occurred. The API will never clear the variable. It is up to the application to clear the variable, when the event has been handled.*
- `typedef u32 vtss_packet_rate_t`

*Policer packet rate in PPS.*
- `typedef u32 vtss_port_no_t`

*Port Number.*
- `typedef u32 vtss_phys_port_no_t`

*Physical port number.*
- `typedef u32 vtss_prio_t`

- **typedef u32 vtss\_queue\_t**  
*Priority number.*
- **typedef u32 vtss\_tagprio\_t**  
*Queue number.*
- **typedef u32 vtss\_dei\_t**  
*Tag Priority or Priority Code Point (PCP)*
- **typedef BOOL vtss\_dei\_t**  
*Drop Eligible Indicator (DEI)*
- **typedef u8 vtss\_dp\_level\_t**  
*Drop Precedence Level (DPL)*
- **typedef u8 vtss\_pct\_t**  
*Percentage, 0-100.*
- **typedef u32 vtss\_bitrate\_t**  
*Policer/Shaper bit rate in kbps (1000 bits per second). The rate will be rounded to the nearest value supported by the chip.*
- **typedef u32 vtss\_burst\_level\_t**  
*Policer/shaper burst level in bytes. The level will be rounded to the nearest value supported by the chip.*
- **typedef u8 vtss\_dscp\_t**  
*DSCP value (0-63)*
- **typedef u32 vtss\_qce\_id\_t**  
*QoS Control Entry ID.*
- **typedef u16 vtss\_evc\_policer\_id\_t**  
*EVC policer index.*
- **typedef u32 vtss\_wred\_group\_t**  
*WRED group number.*
- **typedef u16 vtss\_vid\_t**  
*VLAN Identifier.*
- **typedef u16 vtss\_etype\_t**  
*Ethernet Type.*
- **typedef u8 vtss\_mac\_addr\_t[VTSS\_MAC\_ADDR\_SZ\_BYTES]**
- **typedef u16 vtss\_evc\_id\_t**  
*EVC ID.*
- **typedef u32 vtss\_isdx\_t**
- **typedef u32 vtss\_aggr\_no\_t**  
*Aggregation Number.*
- **typedef u32 vtss\_glag\_no\_t**  
*Description: GLAG number.*
- **typedef u32 vtss\_packet\_rx\_queue\_t**  
*Description: CPU Rx queue number.*
- **typedef u32 vtss\_packet\_rx\_grp\_t**  
*Description: CPU Rx group number.*
- **typedef u32 vtss\_packet\_tx\_grp\_t**  
*Description: CPU Tx group number.*
- **typedef u16 vtss\_udp\_tcp\_t**  
*Description: UDP/TCP port number.*
- **typedef u32 vtss\_ip\_t**  
*IPv4 address/mask.*
- **typedef vtss\_ip\_t vtss\_ipv4\_t**  
*IPv4 address/mask.*
- **typedef u32 vtss\_prefix\_size\_t**  
*Prefix size.*
- **typedef u16 vtss\_vcap\_vr\_value\_t**

- VCAP universal value or range type.
- **typedef u32 vtss\_acl\_policer\_no\_t**  
*ACL policer number.*
  - **typedef u32 vtss\_acl\_policy\_no\_t**  
*ACL policy number.*
  - **typedef u32 vtss\_ece\_id\_t**  
*EVC Control Entry (ECE) ID.*
  - **typedef u64 vtss\_counter\_t**  
*Counter.*
  - **typedef u16 vtss\_hqos\_id\_t**  
*HQoS entry identifier (HQoS ID)*
  - **typedef i64 vtss\_clk\_adj\_rate\_t**  
*Clock adjustment rate in parts per billion (ppb) \* 1<<16. Range is +2\*\*47 ppb For example, 8.25 ppb is expressed as 0x0000.0000.0008.4000.*
  - **typedef i64 vtss\_timeinterval\_t**  
*Time interval in ns \* 1<<16 range +2\*\*47 ns = 140737 sec = 39 hours For example, 2.5 ns is expressed as 0x0000.0000.0002.8000.*
  - **typedef u8 vtss\_clock\_identity[VTSS\_CLOCK\_IDENTITY\_LENGTH]**  
*PTP clock unique identifier.*

## Enumerations

- **enum {**
- VTSS\_RC\_OK** = 0, **VTSS\_RC\_ERROR** = -1, **VTSS\_RC\_INV\_STATE** = -2, **VTSS\_RC\_INCOMPLETE** = -3, **VTSS\_RC\_ERR\_CLK\_CONF\_NOT\_SUPPORTED** = -6, **VTSS\_RC\_ERR\_KR\_CONF\_NOT\_SUPPORTED** = -7, **VTSS\_RC\_ERR\_KR\_CONF\_INVALID\_PARAMETER** = -8, **VTSS\_RC\_ERR\_PHY\_BASE\_NO\_NOT\_FOUND** = -50, **VTSS\_RC\_ERR\_PHY\_6G\_MACRO\_SETUP** = -51, **VTSS\_RC\_ERR\_PHY\_MEDIA\_IF\_NOT\_SUPPORTED** = -52, **VTSS\_RC\_ERR\_PHY\_CLK\_CONF\_NOT\_SUPPORTED** = -53, **VTSS\_RC\_ERR\_PHY\_GPIO\_ALT\_MODE\_NOT\_SUPPORTED** = -54, **VTSS\_RC\_ERR\_PHY\_GPIO\_PIN\_NOT\_SUPPORTED** = -55, **VTSS\_RC\_ERR\_PHY\_PORT\_OUT\_RANGE** = -56, **VTSS\_RC\_ERR\_PHY\_PATCH\_SETTING\_NOT\_SUPPORTED** = -57, **VTSS\_RC\_ERR\_PHY\_LCPLL\_NOT\_SUPPORTED** = -58, **VTSS\_RC\_ERR\_PHY\_RCPLL\_NOT\_SUPPORTED** = -59, **VTSS\_RC\_ERR\_MACSEC\_INVALID\_SCI\_MACADDR** = -60, **VTSS\_RC\_ERR\_MACSEC\_NOT\_ENABLED** = -61, **VTSS\_RC\_ERR\_MACSEC\_SECY\_ALREADY\_IN\_USE** = -63, **VTSS\_RC\_ERR\_MACSEC\_NO\_SECY\_FOUND** = -64, **VTSS\_RC\_ERR\_MACSEC\_NO\_SECY\_VACANCY** = -65, **VTSS\_RC\_ERR\_MACSEC\_INVALID\_VALIDATE\_FRM** = -66, **VTSS\_RC\_ERR\_MACSEC\_COULD\_NOT\_PRG\_SA\_MAC** = -67, **VTSS\_RC\_ERR\_MACSEC\_COULD\_NOT\_PRG\_SA\_FLOW** = -68, **VTSS\_RC\_ERR\_MACSEC\_COULD\_NOT\_ENA\_SA** = -69, **VTSS\_RC\_ERR\_MACSEC\_COULD\_NOT\_SET\_SA** = -70, **VTSS\_RC\_ERR\_MACSEC\_INVALID\_BYPASS\_HDR\_LEN** = -71, **VTSS\_RC\_ERR\_MACSEC\_SC\_NOT\_FOUND** = -72, **VTSS\_RC\_ERR\_MACSEC\_NO\_CTRL\_FRM\_MATCH** = -73, **VTSS\_RC\_ERR\_MACSEC\_COULD\_NOT\_SET\_PATTERN** = -74, **VTSS\_RC\_ERR\_MACSEC\_TIMEOUT\_ISSUE** = -75, **VTSS\_RC\_ERR\_MACSEC\_COULD\_NOT\_EMPTY\_EGRESS** = -76, **VTSS\_RC\_ERR\_MACSEC\_AN\_NOT\_CREATED** = -77, **VTSS\_RC\_ERR\_MACSEC\_COULD\_NOT\_EMPTY\_INGRESS** = -78, **VTSS\_RC\_ERR\_MACSEC\_TX\_SC\_NOT\_EXIST** = -80, **VTSS\_RC\_ERR\_MACSEC\_COULD\_NOT\_DISABLE\_SA** = -81, **VTSS\_RC\_ERR\_MACSEC\_COULD\_NOT\_DEL\_RX\_SA** = -82, **VTSS\_RC\_ERR\_MACSEC\_COULD\_NOT\_DEL\_TX\_SA** = -83, **VTSS\_RC\_ERR\_MACSEC\_PATTERN\_NOT\_SET** = -84, **VTSS\_RC\_ERR\_MACSEC\_HW\_RESOURCE\_EXHUSTED** = -85, **VTSS\_RC\_ERR\_MACSEC\_SCI\_ALREADY\_EXISTS** = -86, **VTSS\_RC\_ERR\_MACSEC\_SC\_RESOURCE\_NOT\_FOUND** = -87, **VTSS\_RC\_ERR\_MACSEC\_RX\_AN\_ALREADY\_IN** = -88, **VTSS\_RC\_ERR\_MACSEC\_EMPTY\_RECORD** = -89, **VTSS\_RC\_ERR\_MACSEC\_COULD\_NOT\_PRG\_XFORM**

```
= -90, VTSS_RC_ERR_MACSEC_COULD_NOT_TOGGLE_SA = -91, VTSS_RC_ERR_MACSEC_TX_AN_ALREADY_IN_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_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_CPU_ONLY` }

*Packet registration type.*

- enum `vtss_vdd_t` { `VTSS_VDD_1V0`, `VTSS_VDD_1V2` }

*VDD power supply.*

- enum `vtss_ip_type_t` { `VTSS_IP_TYPE_NONE` = 0, `VTSS_IP_TYPE_IPV4` = 1, `VTSS_IP_TYPE_IPV6` = 2 }

- IP address type.*
- enum `vtss_routing_entry_type_t` { `VTSS_ROUTING_ENTRY_TYPE_INVALID` = 0, `VTSS_ROUTING_ENTRY_TYPE_IPV6_UC` = 1, `VTSS_ROUTING_ENTRY_TYPE_IPV4_MC` = 2, `VTSS_ROUTING_ENTRY_TYPE_IPV4_UC` = 3 }

*Routing entry type.*

  - enum `vtss_vcap_bit_t` { `VTSS_VCAP_BIT_ANY`, `VTSS_VCAP_BIT_0`, `VTSS_VCAP_BIT_1` }

*VCAP 1 bit.*

  - enum `vtss_vcap_vr_type_t` { `VTSS_VCAP_VR_TYPE_VALUE_MASK`, `VTSS_VCAP_VR_TYPE_RANGE_INCLUSIVE`, `VTSS_VCAP_VR_TYPE_RANGE_EXCLUSIVE` }

*Value/Range type.*

  - enum `vtss_vcap_key_type_t` { `VTSS_VCAP_KEY_TYPE_NORMAL`, `VTSS_VCAP_KEY_TYPE_DOUBLE_TAG`, `VTSS_VCAP_KEY_TYPE_IP_ADDR`, `VTSS_VCAP_KEY_TYPE_MAC_IP_ADDR` }

*VCAP key type.*

  - enum `vtss_ece_dir_t` { `VTSS_ECE_DIR_BOTH`, `VTSS_ECE_DIR_UNI_TO_NNI`, `VTSS_ECE_DIR_NNI_TO_UNI` }

*ECE direction.*

  - enum `vtss_ece_pop_tag_t` { `VTSS_ECE_POP_TAG_0`, `VTSS_ECE_POP_TAG_1`, `VTSS_ECE_POP_TAG_2` }

*Ingress tag popping.*

  - enum `vtss_hqos_sch_mode_t` { `VTSS_HQOS_SCH_MODE_NORMAL`, `VTSS_HQOS_SCH_MODE_BASIC`, `VTSS_HQOS_SCH_MODE_HIERARCHICAL` }

*HQoS port scheduling mode.*

## 7.5.1 Detailed Description

Generic types API.

This header file describes generic types used in the API

## 7.5.2 Macro Definition Documentation

### 7.5.2.1 PRIu64

```
#define PRIu64 "llu"
```

Fallback un-signed 64-bit formatting string

Definition at line 111 of file types.h.

### 7.5.2.2 PRIi64

```
#define PRIi64 "lli"
```

Fallback signed 64-bit formatting string

Definition at line 115 of file types.h.

### 7.5.2.3 PRIx64

```
#define PRIx64 "llx"

Fallback hex 64-bit formatting string

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

### 7.5.2.4 VTSS\_BIT64

```
#define VTSS_BIT64(
    x ) (1ULL << (x))

Set one bit in a 64-bit mask

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

### 7.5.2.5 VTSS\_BITMASK64

```
#define VTSS_BITMASK64 (
    x ) ((1ULL << (x)) - 1)

Get a bitmask consisting of x ones

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

### 7.5.2.6 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.
```

### 7.5.2.7 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.
```

### 7.5.2.8 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.

### 7.5.2.9 TRUE

```
#define TRUE 1
```

True boolean value

Definition at line 129 of file types.h.

### 7.5.2.10 FALSE

```
#define FALSE 0
```

False boolean value

Definition at line 132 of file types.h.

### 7.5.2.11 VTSS\_PACKET\_RATE\_DISABLED

```
#define VTSS_PACKET_RATE_DISABLED 0xffffffff
```

Special value for disabling packet policer

Definition at line 238 of file types.h.

### 7.5.2.12 VTSS\_PORT\_COUNT [1/2]

```
#define VTSS_PORT_COUNT 1
```

Default number of ports

Number of ports

Definition at line 336 of file types.h.

### 7.5.2.13 VTSS\_PORT\_COUNT [2/2]

```
#define VTSS_PORT_COUNT 26
```

Default number of ports

Number of ports

Definition at line 336 of file types.h.

### 7.5.2.14 VTSS\_PORTS

```
#define VTSS_PORTS VTSS_OPT_PORT_COUNT
```

Number of ports

Definition at line 442 of file types.h.

### 7.5.2.15 VTSS\_PORT\_NO\_NONE

```
#define VTSS_PORT_NO_NONE (0xffffffff)
```

Port number none

Definition at line 448 of file types.h.

### 7.5.2.16 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.

### 7.5.2.17 VTSS\_PORT\_NO\_START

```
#define VTSS_PORT_NO_START (0)
```

Port start number

Definition at line 450 of file types.h.

### 7.5.2.18 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.

### 7.5.2.19 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.

### 7.5.2.20 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.

### 7.5.2.21 VTSS\_PRIOS

```
#define VTSS_PRIOS 8
```

Number of priorities

Definition at line 515 of file types.h.

### 7.5.2.22 VTSS\_PRIO\_NO\_NONE

```
#define VTSS_PRIO_NO_NONE 0xffffffff
```

Priority number none (= undefined)

Definition at line 516 of file types.h.

### 7.5.2.23 VTSS\_PRIO\_START

```
#define VTSS_PRIO_START 0
```

Priority start number (lowest)

Definition at line 517 of file types.h.

### 7.5.2.24 VTSS\_PRIO\_END

```
#define VTSS_PRIO_END (VTSS_PRIO_START + VTSS_PRIOS)
```

Priority end number

Definition at line 518 of file types.h.

### 7.5.2.25 VTSS\_PRIO\_ARRAY\_SIZE

```
#define VTSS_PRIO_ARRAY_SIZE VTSS_PRIO_END
```

Priority number array size

Definition at line 519 of file types.h.

### 7.5.2.26 VTSS\_QUEUES

```
#define VTSS_QUEUES VTSS_PRIOS
```

Number of queues

Definition at line 523 of file types.h.

### 7.5.2.27 VTSS\_QUEUE\_START

```
#define VTSS_QUEUE_START 0
```

Queue start number

Definition at line 524 of file types.h.

### 7.5.2.28 VTSS\_QUEUE\_END

```
#define VTSS_QUEUE_END (VTSS_QUEUE_START + VTSS_QUEUES)
```

Queue end number

Definition at line 525 of file types.h.

### 7.5.2.29 VTSS\_QUEUE\_ARRAY\_SIZE

```
#define VTSS_QUEUE_ARRAY_SIZE VTSS_QUEUE_END
```

Queue number array size

Definition at line 526 of file types.h.

### 7.5.2.30 VTSS\_PCPS

```
#define VTSS_PCPS 8
```

Number of PCP values

Definition at line 530 of file types.h.

### 7.5.2.31 VTSS\_PCP\_START

```
#define VTSS_PCP_START 0
```

PCP start number

Definition at line 531 of file types.h.

### 7.5.2.32 VTSS\_PCP\_END

```
#define VTSS_PCP_END (VTSS_PCP_START + VTSS_PCPS)
```

PCP end number

Definition at line 532 of file types.h.

### 7.5.2.33 VTSS\_PCP\_ARRAY\_SIZE

```
#define VTSS_PCP_ARRAY_SIZE VTSS_PCP_END
```

PCP array size

Definition at line 533 of file types.h.

### 7.5.2.34 VTSS\_DEIS

```
#define VTSS_DEIS 2
```

Number of DEI values

Definition at line 537 of file types.h.

### 7.5.2.35 VTSS\_DEI\_START

```
#define VTSS_DEI_START 0
```

DEI start number

Definition at line 538 of file types.h.

### 7.5.2.36 VTSS\_DEI\_END

```
#define VTSS_DEI_END (VTSS_DEI_START + VTSS_DEIS)
```

DEI end number

Definition at line 539 of file types.h.

### 7.5.2.37 VTSS\_DEI\_ARRAY\_SIZE

```
#define VTSS_DEI_ARRAY_SIZE VTSS_DEI_END
```

DEI array size

Definition at line 540 of file types.h.

### 7.5.2.38 VTSS\_DPLS

```
#define VTSS_DPLS 2
```

Default number of drop precedence levels

Definition at line 544 of file types.h.

### 7.5.2.39 VTSS\_DPL\_START

```
#define VTSS_DPL_START 0
```

DPL start number

Definition at line 551 of file types.h.

### 7.5.2.40 VTSS\_DPL\_END

```
#define VTSS_DPL_END (VTSS_DPL_START + VTSS_DPLS)
```

DPL end number

Definition at line 552 of file types.h.

### 7.5.2.41 VTSS\_DPL\_ARRAY\_SIZE

```
#define VTSS_DPL_ARRAY_SIZE VTSS_DPL_END
```

DPL array size

Definition at line 553 of file types.h.

### 7.5.2.42 VTSS\_BITRATE\_DISABLED

```
#define VTSS_BITRATE_DISABLED 0xffffffff
```

Bitrate disabled

Definition at line 563 of file types.h.

#### 7.5.2.43 VTSS\_VID\_NULL

```
#define VTSS_VID_NULL ((const vtss_vid_t)0)
```

NULL VLAN ID

Definition at line 609 of file types.h.

#### 7.5.2.44 VTSS\_VID\_DEFAULT

```
#define VTSS_VID_DEFAULT ((const vtss_vid_t)1)
```

Default VLAN ID

Definition at line 610 of file types.h.

#### 7.5.2.45 VTSS\_VID\_RESERVED

```
#define VTSS_VID_RESERVED ((const vtss_vid_t)0xFFFF)
```

Reserved VLAN ID

Definition at line 611 of file types.h.

#### 7.5.2.46 VTSS\_VIDS

```
#define VTSS_VIDS ((const vtss_vid_t)4096)
```

Number of VLAN IDs

Definition at line 612 of file types.h.

#### 7.5.2.47 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.

### 7.5.2.48 VTSS\_ETYPE\_VTSS

```
#define VTSS_ETYPE_VTSS 0x8880
```

Vitesse Ethernet Type

Definition at line 640 of file types.h.

### 7.5.2.49 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.

### 7.5.2.50 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.

### 7.5.2.51 VTSS\_EVCS

```
#define VTSS_EVCS 256
```

Maximum number of Ethernet Virtual Connections

Definition at line 668 of file types.h.

### 7.5.2.52 VTSS\_ISDX\_NONE

```
#define VTSS_ISDX_NONE (0)
```

Ingress Service Index number none

Definition at line 674 of file types.h.

### 7.5.2.53 VTSS\_AGGRS

```
#define VTSS_AGGRS (VTSS_PORTS/2)
```

Number of LLAGs

Definition at line 678 of file types.h.

### 7.5.2.54 VTSS\_AGGR\_NO\_NONE

```
#define VTSS_AGGR_NO_NONE 0xffffffff
```

Aggregation number none

Definition at line 679 of file types.h.

### 7.5.2.55 VTSS\_AGGR\_NO\_START

```
#define VTSS_AGGR_NO_START 0
```

Aggregation start number

Definition at line 680 of file types.h.

### 7.5.2.56 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.

### 7.5.2.57 VTSS\_GLAGS

```
#define VTSS_GLAGS 2
```

Number of GLAGs

Definition at line 689 of file types.h.

### 7.5.2.58 VTSS\_GLAG\_NO\_NONE

```
#define VTSS_GLAG_NO_NONE 0xffffffff
```

GLAG number none

Definition at line 691 of file types.h.

### 7.5.2.59 VTSS\_GLAG\_NO\_START

```
#define VTSS_GLAG_NO_START 0
```

GLAG start number

Definition at line 692 of file types.h.

### 7.5.2.60 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.

### 7.5.2.61 VTSS\_GLAG\_PORTS

```
#define VTSS_GLAG_PORTS 8
```

Number of GLAG ports

Definition at line 696 of file types.h.

### 7.5.2.62 VTSS\_GLAG\_PORT\_START

```
#define VTSS_GLAG_PORT_START 0
```

GLAG port start number

Definition at line 697 of file types.h.

### 7.5.2.63 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.

### 7.5.2.64 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.

### 7.5.2.65 VTSS\_PACKET\_RX\_QUEUE\_CNT

```
#define VTSS_PACKET_RX_QUEUE_CNT 8
```

Number of Rx packet queues

Definition at line 720 of file types.h.

### 7.5.2.66 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 722 of file types.h.

### 7.5.2.67 VTSS\_PACKET\_TX\_GRP\_CNT

```
#define VTSS_PACKET_TX_GRP_CNT 2
```

Number of Tx packet groups

Definition at line 724 of file types.h.

### 7.5.2.68 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.

### 7.5.2.69 VTSS\_PACKET\_RX\_QUEUE\_START

```
#define VTSS_PACKET_RX_QUEUE_START (0)
```

Rx queue start number

Definition at line 746 of file types.h.

### 7.5.2.70 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.

### 7.5.2.71 VTSS\_ACL\_POLICERS

```
#define VTSS_ACL_POLICERS 16
```

Number of ACL policers

Definition at line 1028 of file types.h.

### 7.5.2.72 VTSS\_ACL\_POLICER\_NO\_START

```
#define VTSS_ACL_POLICER_NO_START 0
```

ACL policer start number

Definition at line 1029 of file types.h.

### 7.5.2.73 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.

### 7.5.2.74 VTSS\_ACL\_POLICY\_NO\_NONE

```
#define VTSS_ACL_POLICY_NO_NONE 0xffffffff
```

ACLs disabled on port

Definition at line 1034 of file types.h.

### 7.5.2.75 VTSS\_ACL\_POLICY\_NO\_MIN

```
#define VTSS_ACL_POLICY_NO_MIN 0
```

ACLs policy minimum number

Definition at line 1035 of file types.h.

### 7.5.2.76 VTSS\_ACL\_POLICY\_NO\_MAX

```
#define VTSS_ACL_POLICY_NO_MAX 255
```

ACLs policy maximum number

Definition at line 1037 of file types.h.

### 7.5.2.77 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.

### 7.5.2.78 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.

### 7.5.2.79 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.

### 7.5.2.80 VTSS\_HQOS\_COUNT

```
#define VTSS_HQOS_COUNT 256
```

Maximum number of HQoS entries

Definition at line 1165 of file types.h.

### 7.5.2.81 VTSS\_HQOS\_ID\_NONE

```
#define VTSS_HQOS_ID_NONE 0xffff
```

Special HQoS ID value

Definition at line 1167 of file types.h.

### 7.5.2.82 VTSS\_ONE\_MIA

```
#define VTSS_ONE_MIA 1000000000
```

One billion

Definition at line 1198 of file types.h.

### 7.5.2.83 VTSS\_ONE\_MILL

```
#define VTSS_ONE_MILL 1000000
```

One million

Definition at line 1199 of file types.h.

### 7.5.2.84 VTSS\_MAX\_TIMEINTERVAL

```
#define VTSS_MAX_TIMEINTERVAL 0x7fffffffffffffLL
```

Maximum time interval

Definition at line 1200 of file types.h.

### 7.5.2.85 VTSS\_INTERVAL\_SEC

```
#define VTSS_INTERVAL_SEC( t ) ((i32)VTSS_DIV64((t)>>16, VTSS_ONE_MILL))
```

One Second time interval

Definition at line 1202 of file types.h.

### 7.5.2.86 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.

### 7.5.2.87 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.

### 7.5.2.88 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.

### 7.5.2.89 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.

### 7.5.2.90 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.

### 7.5.2.91 VTSS\_CLOCK\_IDENTITY\_LENGTH

```
#define VTSS_CLOCK_IDENTITY_LENGTH 8
```

Length of unique PTP identifier

Definition at line 1218 of file types.h.

### 7.5.2.92 VTSS\_SYNCE\_CLK\_PORT\_ARRAY\_SIZE

```
#define VTSS_SYNCE_CLK_PORT_ARRAY_SIZE 2
```

SYNCE clock out port numberarray size

Definition at line 1232 of file types.h.

### 7.5.3 Typedef Documentation

#### 7.5.3.1 i8

```
typedef char i8
```

Fallback Integer types.

8-bit signed

Definition at line 68 of file types.h.

#### 7.5.3.2 i16

```
typedef signed short i16
```

16-bit signed

Definition at line 69 of file types.h.

#### 7.5.3.3 i32

```
typedef signed int i32
```

32-bit signed

Definition at line 70 of file types.h.

#### 7.5.3.4 i64

```
typedef signed long long i64
```

64-bit signed

Definition at line 71 of file types.h.

### 7.5.3.5 u8

```
typedef unsigned char u8
```

8-bit unsigned

Definition at line 73 of file types.h.

### 7.5.3.6 u16

```
typedef unsigned short u16
```

16-bit unsigned

Definition at line 74 of file types.h.

### 7.5.3.7 u32

```
typedef unsigned int u32
```

32-bit unsigned

Definition at line 75 of file types.h.

### 7.5.3.8 u64

```
typedef unsigned long long u64
```

64-bit unsigned

Definition at line 76 of file types.h.

### 7.5.3.9 BOOL

```
typedef unsigned char BOOL
```

Boolean implemented as 8-bit unsigned

Definition at line 78 of file types.h.

### 7.5.3.10 uintptr\_t

```
typedef unsigned int uintptr_t
```

Unsigned integer big enough to hold pointers

Definition at line 79 of file types.h.

### 7.5.3.11 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.

### 7.5.3.12 vtss\_isdx\_t

```
typedef u32 vtss_isdx_t
```

Ingress Service Index type

Definition at line 673 of file types.h.

### 7.5.3.13 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.

### 7.5.3.14 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.

## 7.5.4 Enumeration Type Documentation

### 7.5.4.1 anonymous enum

```
anonymous enum
```

Error codes.

## Enumerator

VTSS_RC_OK	Success
VTSS_RC_ERROR	Unspecified error
VTSS_RC_INV_STATE	Invalid state for operation
VTSS_RC_INCOMPLETE	Incomplete result
VTSS_RC_ERR_CLK_CONF_NOT_SUPPORTED	The PHY doesn't support clock configuration (for SyncE)
VTSS_RC_ERR_KR_CONF_NOT_SUPPORTED	The PHY doesn't support 10GBASE_KR equalization
VTSS_RC_ERR_KR_CONF_INVALID_PARAMETER	One of the parameters are out of range
VTSS_RC_ERR_PHY_BASE_NO_NOT_FOUND	Port base number (first port within a chip) is not found
VTSS_RC_ERR_PHY_6G_MACRO_SETUP	Setup of 6G macro failed
VTSS_RC_ERR_PHY_MEDIA_IF_NOT_SUPPORTED	PHY does not support the selected media mode
VTSS_RC_ERR_PHY_CLK_CONF_NOT_SUPPORTED	The PHY doesn't support clock configuration (for SyncE)
VTSS_RC_ERR_PHY_GPIO_ALT_MODE_NOT_SUPPORTED	The PHY doesn't support the alternative mode for the selected GPIO pin
VTSS_RC_ERR_PHY_GPIO_PIN_NOT_SUPPORTED	The PHY doesn't support the selected GPIO pin
VTSS_RC_ERR_PHY_PORT_OUT_RANGE	PHY API called with port number larger than VTSS_PORTS
VTSS_RC_ERR_PHY_PATCH_SETTING_NOT_SUPPORTED	PHY API micro patch setting not supported for the port in question
VTSS_RC_ERR_PHY_LCPLL_NOT_SUPPORTED	PHY API LC-PLL status not supported for the port
VTSS_RC_ERR_PHY_RCPLL_NOT_SUPPORTED	PHY API RC-PLL status not supported for the port
VTSS_RC_ERR_MACSEC_INVALID_SCI_MACADDR	From IEEE 802.1AE-2006, section 9.9 - The 64-bit value FF-FF-FF-FF-FF-FF is never used as an SCI and is reserved for use by implementations to indicate the absence of an SC or an SCI in contexts where an SC can be present
VTSS_RC_ERR_MACSEC_NOT_ENABLED	Trying to access port where MACSEC is not enabled
VTSS_RC_ERR_MACSEC_SECY_ALREADY_IN_USE	Trying to use a secy which is already in use
VTSS_RC_ERR_MACSEC_NO_SECY_FOUND	No SecY found for the specific port
VTSS_RC_ERR_MACSEC_NO_SECY_VACANCY	No secy vacancy
VTSS_RC_ERR_MACSEC_INVALID_VALIDATE_FRAME	Validate_frames value invalid
VTSS_RC_ERR_MACSEC_COULD_NOT_PRG_SA_MATCH	Could not program the SA match
VTSS_RC_ERR_MACSEC_COULD_NOT_PRG_SA_FLOW	Could not program the SA flow
VTSS_RC_ERR_MACSEC_COULD_NOT_ENA_SA	Could not enable the SA
VTSS_RC_ERR_MACSEC_COULD_NOT_SET_SA	Could not set SA to in use
VTSS_RC_ERR_MACSEC_INVALID_BYPASS_HEADER_LENGTH	Invalid header bypass length
VTSS_RC_ERR_MACSEC_SC_NOT_FOUND	Could not find SC (from sci)
VTSS_RC_ERR_MACSEC_NO_CTRL_FRAME_MATCH	No control frame match
VTSS_RC_ERR_MACSEC_COULD_NOT_SET_PATTERN	Could no set bypass pattern for CP rule
VTSS_RC_ERR_MACSEC_TIMEOUT_ISSUE	Internal timeout issue, bailing out

## Enumerator

VTSS_RC_ERR_MACSEC_COULD_NOT_EMPT← Y_EGRESS	Could not empty the egress pipeline
VTSS_RC_ERR_MACSEC_AN_NOT_CREATED	AN not created.
VTSS_RC_ERR_MACSEC_COULD_NOT_EMPT← Y_INGRESS	Could not empty the ingress pipeline
VTSS_RC_ERR_MACSEC_TX_SC_NOT_EXIST	No tx SC found
VTSS_RC_ERR_MACSEC_COULD_NOT_DISAB← LE_SA	Could not disable sa
VTSS_RC_ERR_MACSEC_COULD_NOT_DEL_R← X_SA	Could not delete rx sa
VTSS_RC_ERR_MACSEC_COULD_NOT_DEL_T← X_SA	Could not delete tx sa
VTSS_RC_ERR_MACSEC_PATTERN_NOT_SET	Pattern not set
VTSS_RC_ERR_MACSEC_HW_RESOURCE_EX← HUSTED	HW resources exhausted
VTSS_RC_ERR_MACSEC_SCI_ALREADY_EXISTS	SCI already exists
VTSS_RC_ERR_MACSEC_SC_RESOURCE_NO← T_FOUND	Could not find SC resources
VTSS_RC_ERR_MACSEC_RX_AN_ALREADY_I← N_USE	Rx AN is in use
VTSS_RC_ERR_MACSEC_EMPTY_RECORD	Could not get an empty record
VTSS_RC_ERR_MACSEC_COULD_NOT_PRG← XFORM	Could not program the xform record
VTSS_RC_ERR_MACSEC_COULD_NOT_TOGG← LE_SA	Could not toggle SA
VTSS_RC_ERR_MACSEC_TX_AN_ALREADY_I← N_USE	Tx AN is in use
VTSS_RC_ERR_MACSEC_ALL_AVAILABLE_SA← _IN_USE	All available SA's are in use
VTSS_RC_ERR_MACSEC_MATCH_DISABLE	MACSEC match disabled
VTSS_RC_ERR_MACSEC_ALL_CP_RULES_IN← USE	All CP rules of the specific type are in use
VTSS_RC_ERR_MACSEC_PATTERN_PRIO_NO← T_VALID	The pattern priority is not valid
VTSS_RC_ERR_MACSEC_BUFFER_TOO_SMALL	Buffer to small, must be greater than VTSS_MACSEC_FRAME_CAPTURE_SIZE_MAX
VTSS_RC_ERR_MACSEC_FRAME_TOO_LONG	Frame length is supposed to be less than the amount of data in the fifo
VTSS_RC_ERR_MACSEC_FRAME_TRUNCATED	Frame is Truncated
VTSS_RC_ERR_MACSEC_PHY_POWERED_DO← WN	Phy is powered down, i.e. the MacSec block is not accessible
VTSS_RC_ERR_MACSEC_PHY_NOT_MACSEC← _CAPABLE	Port/Phy is not MacSec capable
VTSS_RC_ERR_MACSEC_AN_NOT_EXIST	AN does not exist
VTSS_RC_ERR_MACSEC_NO_PATTERN_CFG	No pattern is configured
VTSS_RC_ERR_MACSEC_MAX_MTU	Maximum MTU allowed is 32761 (+ 4 bytes for VLAN)
VTSS_RC_ERR_MACSEC_UNEXPECT_CP_MODE	Unexpected CP mode
VTSS_RC_ERR_MACSEC_COULD_NOT_DISAB← LE_AN	Could not disable AN
VTSS_RC_ERR_MACSEC_RULE_OUT_OF_RAN← GE	Rule id is out of range. Must not be larger than VTSS_MACSEC_CP_RULES

**Enumerator**

VTSS_RC_ERR_MACSEC_RULE_NOT_EXIST	Rule does not exist
VTSS_RC_ERR_MACSEC_CSR_READ	Could not do CSR read
VTSS_RC_ERR_MACSEC_CSR_WRITE	Could not do CSR write
VTSS_RC_ERR_PHY_6G_RCPLL_ON_BASE_P←_ORT_ONLY	PHY API 6G RC-PLL status support only on Base port
VTSS_RC_ERR_INVALID_NULL_PTR	A pointer was unexpected NULL

Definition at line 139 of file types.h.

**7.5.4.2 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.

**7.5.4.3 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.

## 7.5.4.4 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.

#### 7.5.4.5 `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.

#### 7.5.4.6 `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.

#### 7.5.4.7 `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.

#### 7.5.4.8 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_CPU_ONLY	Redirect to CPU

Definition at line 753 of file types.h.

#### 7.5.4.9 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.

#### 7.5.4.10 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.

#### 7.5.4.11 `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.

#### 7.5.4.12 `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.

#### 7.5.4.13 `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.

## 7.5.4.14 vtss\_ece\_dir\_t

enum `vtss_ece_dir_t`

ECE direction.

Enumerator

VTSS_ECE_DIR_BOTH	Bidirectional
VTSS_ECE_DIR_UNI_TO_NNI	UNI-to-NNI direction
VTSS_ECE_DIR_NNI_TO_UNI	NNI-to-UNI direction

Definition at line 1058 of file types.h.

## 7.5.4.15 vtss\_ece\_pop\_tag\_t

enum `vtss_ece_pop_tag_t`

Ingress tag popping.

Enumerator

VTSS_ECE_POP_TAG_0	No tag popping
VTSS_ECE_POP_TAG_1	Pop one tag
VTSS_ECE_POP_TAG_2	Pop two tags (VTSS_ECE_DIR_NNI_TO_UNI only)

Definition at line 1066 of file types.h.

## 7.5.4.16 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.

## 7.6 vtss\_api/include/vtss\_ae\_api.h File Reference

ae API

```
#include <vtss/api/types.h>
```

### 7.6.1 Detailed Description

ae API

## 7.7 vtss\_api/include/vtss\_afi\_api.h File Reference

AFI API.

```
#include <vtss/api/options.h>
```

### 7.7.1 Detailed Description

AFI API.

This header file describes Automatic Frame Injector functions.

## 7.8 vtss\_api/include/vtss\_aneg\_api.h File Reference

ANEG API.

```
#include <vtss/api/types.h>
```

### 7.8.1 Detailed Description

ANEG API.

## 7.9 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/qos_api.h>
#include <vtss/packet_api.h>
#include <vtss/security_api.h>
#include <vtss/l2_api.h>
#include <vtss/evc_api.h>
#include <vtss/sync_api.h>
#include <vtss/phy_ts_api.h>
#include <vtss/ts_api.h>
```

### 7.9.1 Detailed Description

Vitesse API main header file.

This is the only header file which must be included by the application

## 7.10 vtss\_api/include/vtss\_evc\_api.h File Reference

EVC API.

```
#include <vtss/api/types.h>
```

### Data Structures

- struct [vtss\\_evc\\_port\\_conf\\_t](#)  
*EVC port configuration.*
- struct [vtss\\_evc\\_inner\\_tag\\_t](#)  
*EVC inner tag.*
- struct [vtss\\_evc\\_pb\\_conf\\_t](#)  
*PB specific EVC configuration.*
- struct [vtss\\_evc\\_conf\\_t](#)  
*EVC configuration (excluding UNIs)*
- struct [vtss\\_ece\\_mac\\_t](#)  
*ECE MAC information.*
- struct [vtss\\_ece\\_tag\\_t](#)  
*ECE tag information.*
- struct [vtss\\_ece\\_frame\\_ipv4\\_t](#)  
*ECE IPv4 information.*

- struct `vtss_ece_frame_ipv6_t`  
*ECE IPv6 information.*
- struct `vtss_ece_key_t`  
*ECE key.*
- struct `vtss_ece_outer_tag_t`  
*ECE outer tag.*
- struct `vtss_ece_action_t`  
*ECE action.*
- struct `vtss_ece_t`  
*EVC Control Entry.*
- struct `vtss_mce_key_t`  
*MCE key.*
- struct `vtss_mce_action_t`  
*MCE action.*
- struct `vtss_mce_t`  
*MEP Control Entry.*

## Macros

- `#define VTSS_EVC_POLICERS 256`
- `#define VTSS_EVC_ID_NONE 0xffff`
- `#define VTSS_ECE_ID_LAST 0`
- `#define VTSS_MCE_ID_LAST 0`
- `#define VTSS_MCE_POP_NONE 0xFF`

## Typedefs

- `typedef vtss_dlb_policer_conf_t vtss_evc_policer_conf_t`  
*EVC policer configuration.*
- `typedef u32 vtss_mce_id_t`  
*MEP Control Entry (MCE) ID.*

## Enumerations

- enum `vtss_evc_vid_mode_t` { `VTSS_EVC_VID_MODE_NORMAL`, `VTSS_EVC_VID_MODE_TUNNEL` }  
*EVC VID mode.*
- enum `vtss_evc_inner_tag_type_t` { `VTSS_EVC_INNER_TAG_NONE`, `VTSS_EVC_INNER_TAG_C`, `VTSS_EVC_INNER_TAG_S`, `VTSS_EVC_INNER_TAG_S_CUSTOM` }  
*EVC inner tag type.*
- enum `vtss_ece_type_t` { `VTSS_ECE_TYPE_ANY`, `VTSS_ECE_TYPE_IPV4`, `VTSS_ECE_TYPE_IPV6` }  
*ECE frame type.*
- enum `vtss_ece_port_t` { `VTSS_ECE_PORT_NONE`, `VTSS_ECE_PORT_ROOT` }  
*ECE port type.*

## Functions

- `vtss_rc vtss_evc_port_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_evc_port_conf_t` \*const conf)
 

*Get EVC port configuration.*
- `vtss_rc vtss_evc_port_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_evc_port_conf_t` \*const conf)
 

*Set EVC port configuration.*
- `vtss_rc vtss_evc_policer_conf_get` (const `vtss_inst_t` inst, const `vtss_evc_policer_id_t` policer\_id, `vtss_evc_policer_conf_t` \*const conf)
 

*Get EVC policer configuration.*
- `vtss_rc vtss_evc_policer_conf_set` (const `vtss_inst_t` inst, const `vtss_evc_policer_id_t` policer\_id, const `vtss_evc_policer_conf_t` \*const conf)
 

*Set EVC policer configuration.*
- `vtss_rc vtss_evc_add` (const `vtss_inst_t` inst, const `vtss_evc_id_t` evc\_id, const `vtss_evc_conf_t` \*const conf)
 

*Add EVC.*
- `vtss_rc vtss_evc_del` (const `vtss_inst_t` inst, const `vtss_evc_id_t` evc\_id)
 

*Delete EVC.*
- `vtss_rc vtss_evc_get` (const `vtss_inst_t` inst, const `vtss_evc_id_t` evc\_id, `vtss_evc_conf_t` \*const conf)
 

*Get EVC configuration.*
- `vtss_rc vtss_ece_init` (const `vtss_inst_t` inst, const `vtss_ece_type_t` type, `vtss_ece_t` \*const ece)
 

*Initialize ECE to default values.*
- `vtss_rc vtss_ece_add` (const `vtss_inst_t` inst, const `vtss_ece_id_t` ece\_id, const `vtss_ece_t` \*const ece)
 

*Add/modify ECE.*
- `vtss_rc vtss_ece_del` (const `vtss_inst_t` inst, const `vtss_ece_id_t` ece\_id)
 

*Delete ECE.*
- `vtss_rc vtss_mce_init` (const `vtss_inst_t` inst, `vtss_mce_t` \*const mce)
 

*Initialize MCE to default values.*
- `vtss_rc vtss_mce_add` (const `vtss_inst_t` inst, const `vtss_mce_id_t` mce\_id, const `vtss_mce_t` \*const mce)
 

*Add/modify MCE.*
- `vtss_rc vtss_mce_del` (const `vtss_inst_t` inst, const `vtss_mce_id_t` mce\_id)
 

*Delete MCE.*

### 7.10.1 Detailed Description

EVC API.

This header file describes EVC functions

### 7.10.2 Macro Definition Documentation

#### 7.10.2.1 VTSS\_EVC\_POLICERS

```
#define VTSS_EVC_POLICERS 256
```

Maximum number of EVC policers

Definition at line 199 of file vtss\_evc\_api.h.

### 7.10.2.2 VTSS\_EVC\_ID\_NONE

```
#define VTSS_EVC_ID_NONE 0xffff
```

Special EVC ID value

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

### 7.10.2.3 VTSS\_ECE\_ID\_LAST

```
#define VTSS_ECE_ID_LAST 0
```

Special value used to add last in list

Definition at line 363 of file vtss\_evc\_api.h.

### 7.10.2.4 VTSS\_MCE\_ID\_LAST

```
#define VTSS_MCE_ID_LAST 0
```

Special value used to add last in list

Definition at line 667 of file vtss\_evc\_api.h.

### 7.10.2.5 VTSS\_MCE\_POP\_NONE

```
#define VTSS_MCE_POP_NONE 0xFF
```

Special value used to indicate pop\_cnt not enabled

Definition at line 756 of file vtss\_evc\_api.h.

## 7.10.3 Enumeration Type Documentation

### 7.10.3.1 vtss\_evc\_vid\_mode\_t

```
enum vtss_evc_vid_mode_t
```

EVC VID mode.

**Enumerator**

VTSS_EVC_VID_MODE_NORMAL	Outer VID identifies EVC
VTSS_EVC_VID_MODE_TUNNEL	Inner VID identifies EVC

Definition at line 247 of file vtss\_evc\_api.h.

**7.10.3.2 vtss\_evc\_inner\_tag\_type\_t**

```
enum vtss_evc_inner_tag_type_t
```

EVC inner tag type.

**Enumerator**

VTSS_EVC_INNER_TAG_NONE	No inner tag
VTSS_EVC_INNER_TAG_C	Inner tag is C-tag
VTSS_EVC_INNER_TAG_S	Inner tag is S-tag
VTSS_EVC_INNER_TAG_S_CUSTOM	Inner tag is S-custom tag

Definition at line 254 of file vtss\_evc\_api.h.

**7.10.3.3 vtss\_ece\_type\_t**

```
enum vtss_ece_type_t
```

ECE frame type.

**Enumerator**

VTSS_ECE_TYPE_ANY	Any frame type
VTSS_ECE_TYPE_IPV4	IPv4
VTSS_ECE_TYPE_IPV6	IPv6

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

**7.10.3.4 vtss\_ece\_port\_t**

```
enum vtss_ece_port_t
```

ECE port type.

## Enumerator

VTSS_ECE_PORT_NONE	Port not included
VTSS_ECE_PORT_ROOT	Root UNI port

Definition at line 413 of file vtss\_evc\_api.h.

#### 7.10.4 Function Documentation

##### 7.10.4.1 vtss\_evc\_port\_conf\_get()

```
vtss_rc vtss_evc_port_conf_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_evc_port_conf_t *const conf )
```

Get EVC port configuration.

###### Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>conf</i>	[OUT] EVC port configuration structure.

###### Returns

Return code.

##### 7.10.4.2 vtss\_evc\_port\_conf\_set()

```
vtss_rc vtss_evc_port_conf_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_evc_port_conf_t *const conf )
```

Set EVC port configuration.

###### Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>conf</i>	[IN] EVC port configuration structure.

**Returns**

Return code.

**7.10.4.3 vtss\_evc\_policer\_conf\_get()**

```
vtss_rc vtss_evc_policer_conf_get (
    const vtss_inst_t inst,
    const vtss_evc_policer_id_t policer_id,
    vtss_evc_policer_conf_t *const conf )
```

Get EVC policer configuration.

**Parameters**

<i>inst</i>	[IN] Target instance reference.
<i>policer_id</i>	[IN] Policer ID.
<i>conf</i>	[OUT] Policer configuration.

**Returns**

Return code.

**7.10.4.4 vtss\_evc\_policer\_conf\_set()**

```
vtss_rc vtss_evc_policer_conf_set (
    const vtss_inst_t inst,
    const vtss_evc_policer_id_t policer_id,
    const vtss_evc_policer_conf_t *const conf )
```

Set EVC policer configuration.

**Parameters**

<i>inst</i>	[IN] Target instance reference.
<i>policer_id</i>	[IN] Policer ID.
<i>conf</i>	[IN] Policer configuration.

**Returns**

Return code.

#### 7.10.4.5 vtss\_evc\_add()

```
vtss_rc vtss_evc_add (
    const vtss_inst_t inst,
    const vtss_evc_id_t evc_id,
    const vtss_evc_conf_t *const conf )
```

Add EVC.

##### Parameters

<i>inst</i>	[IN] Target instance reference.
<i>evc_id</i>	[IN] EVC ID.
<i>conf</i>	[IN] EVC configuration.

##### Returns

Return code.

#### 7.10.4.6 vtss\_evc\_del()

```
vtss_rc vtss_evc_del (
    const vtss_inst_t inst,
    const vtss_evc_id_t evc_id )
```

Delete EVC.

##### Parameters

<i>inst</i>	[IN] Target instance reference.
<i>evc_id</i>	[IN] EVC ID.

##### Returns

Return code.

#### 7.10.4.7 vtss\_evc\_get()

```
vtss_rc vtss_evc_get (
    const vtss_inst_t inst,
    const vtss_evc_id_t evc_id,
    vtss_evc_conf_t *const conf )
```

Get EVC configuration.

**Parameters**

<i>inst</i>	[IN] Target instance reference.
<i>evc_id</i>	[IN] EVC ID.
<i>conf</i>	[OUT] EVC configuration.

**Returns**

Return code.

**7.10.4.8 vtss\_ece\_init()**

```
vtss_rc vtss_ece_init (
    const vtss_inst_t inst,
    const vtss_ece_type_t type,
    vtss_ece_t *const ece )
```

Initialize ECE to default values.

**Parameters**

<i>inst</i>	[IN] Target instance reference.
<i>type</i>	[IN] ECE type.
<i>ece</i>	[OUT] ECE structure.

**Returns**

Return code.

**7.10.4.9 vtss\_ece\_add()**

```
vtss_rc vtss_ece_add (
    const vtss_inst_t inst,
    const vtss_ece_id_t ece_id,
    const vtss_ece_t *const ece )
```

Add/modify ECE.

**Parameters**

<i>inst</i>	[IN] Target instance reference.
<i>ece_id</i>	[IN] ECE ID. The ECE will be added before the entry with this ID. VTSS_ECE_ID_LAST is reserved for inserting last.
<i>ece</i>	[IN] ECE structure.

**Returns**

Return code.

**7.10.4.10 vtss\_ece\_del()**

```
vtss_rc vtss_ece_del (
    const vtss_inst_t inst,
    const vtss_ece_id_t ece_id )
```

Delete ECE.

**Parameters**

<i>inst</i>	[IN] Target instance reference.
<i>ece_id</i>	[IN] ECE ID.

**Returns**

Return code.

**7.10.4.11 vtss\_mce\_init()**

```
vtss_rc vtss_mce_init (
    const vtss_inst_t inst,
    vtss_mce_t *const mce )
```

Initialize MCE to default values.

**Parameters**

<i>inst</i>	[IN] Target instance reference.
<i>mce</i>	[OUT] MCE structure.

**Returns**

Return code.

**7.10.4.12 vtss\_mce\_add()**

```
vtss_rc vtss_mce_add (
    const vtss_inst_t inst,
```

```
const vtss_mce_id_t mce_id,
const vtss_mce_t *const mce )
```

Add/modify MCE.

#### Parameters

<i>inst</i>	[IN] Target instance reference.
<i>mce_id</i>	[IN] MCE ID. The MCE will be added before the entry with this ID. VTSS_MCE_ID_LAST is reserved for inserting last.
<i>mce</i>	[IN] MCE structure.

#### Returns

Return code.

### 7.10.4.13 vtss\_mce\_del()

```
vtss_rc vtss_mce_del (
    const vtss_inst_t inst,
    const vtss_mce_id_t mce_id )
```

Delete MCE.

#### Parameters

<i>inst</i>	[IN] Target instance reference.
<i>mce_id</i>	[IN] MCE ID.

#### Returns

Return code.

## 7.11 vtss\_api/include/vtss\_fdma\_api.h File Reference

Frame DMA API.

```
#include <vtss/api/types.h>
```

### 7.11.1 Detailed Description

Frame DMA API.

## 7.12 vtss\_api/include/vtss\_gfp\_api.h File Reference

GFP API.

```
#include <vtss/api/types.h>
```

### 7.12.1 Detailed Description

GFP API.

## 7.13 vtss\_api/include/vtss\_hqos\_api.h File Reference

HQoS API.

```
#include <vtss/api/types.h>
```

### 7.13.1 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.

## 7.14 vtss\_api/include/vtss\_i2c\_api.h File Reference

I2C API.

```
#include <vtss/api/types.h>
```

### 7.14.1 Detailed Description

I2C API.

## 7.15 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
 = 0x7424, VTSS\_TARGET\_CARACAL\_1 = 0x7428,
   
VTSS\_TARGET\_CARACAL\_2 = 0x7429, VTSS\_TARGET\_JAGUAR\_1 = 0x7460, VTSS\_TARGET\_LYNX\_1
 = 0x7462, VTSS\_TARGET\_E\_STAX\_III\_48 = 0x7432,
   
VTSS\_TARGET\_E\_STAX\_III\_68 = 0x7434, VTSS\_TARGET\_E\_STAX\_III\_24\_DUAL = 0xD7431,
   
VTSS\_TARGET\_E\_STAX\_III\_68\_DUAL = 0xD7434, VTSS\_TARGET\_DAYTONA = 0x8492,
   
VTSS\_TARGET\_TALLADEGA = 0x8494, VTSS\_TARGET\_SERVAL\_2 = 0x7438, VTSS\_TARGET\_LYNX\_2
 = 0x7464, VTSS\_TARGET\_JAGUAR\_2 = 0x7468,
   
VTSS\_TARGET\_SPARX\_IV\_52 = 0x7442, VTSS\_TARGET\_SPARX\_IV\_44 = 0x7444, VTSS\_TARGET\_SPARX\_IV\_80
 = 0x7448, VTSS\_TARGET\_SPARX\_IV\_90 = 0x7449 }

*Target chip type.*

- enum **vtss\_pi\_width\_t** { **VTSS\_PI\_WIDTH\_16** = 0, **VTSS\_PI\_WIDTH\_8** }
- PI data width.*

- enum **vtss\_port\_mux\_mode\_t** { **VTSS\_PORT\_MUX\_MODE\_0**, **VTSS\_PORT\_MUX\_MODE\_1**, **VTSS\_PORT\_MUX\_MODE\_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)

- `vtss_rc vtss_inst_destroy` (const `vtss_inst_t` inst)  
*Create target instance.*
- `vtss_rc vtss_init_conf_get` (const `vtss_inst_t` inst, `vtss_init_conf_t` \*const conf)  
*Destroy target instance.*
- `vtss_rc vtss_init_conf_set` (const `vtss_inst_t` inst, const `vtss_init_conf_t` \*const conf)  
*Get default initialization configuration.*
- `vtss_rc vtss_restart_conf_end` (const `vtss_inst_t` inst)  
*Set initialization configuration.*
- `vtss_rc vtss_restart_status_get` (const `vtss_inst_t` inst, `vtss_restart_status_t` \*const status)  
*Indicate configuration end. If a warm start has been done, the stored configuration will be applied.*
- `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)*

### 7.15.1 Detailed Description

Initialization API.

This header file describes functions used to create and initialize targets

### 7.15.2 Macro Definition Documentation

#### 7.15.2.1 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.

### 7.15.3 Typedef Documentation

#### 7.15.3.1 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.

**7.15.3.2 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.

**7.15.3.3 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.

**7.15.3.4 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.

**7.15.3.5 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.

**7.15.3.6 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.

**7.15.3.7 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.

**7.15.3.8 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.

**7.15.3.9 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.

### 7.15.3.10 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.

### 7.15.3.11 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.

### 7.15.3.12 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.

## 7.15.4 Enumeration Type Documentation

### 7.15.4.1 vtss\_target\_type\_t

```
enum vtss_target_type_t
```

Target chip type.

#### Enumerator

VTSS_TARGET_CU_PHY	Cu PHY family
VTSS_TARGET_10G_PHY	10G PHY family
VTSS_TARGET_SPARX_III_11	SparX-III-11 SME switch
VTSS_TARGET_SERVAL_LITE	Serval Lite CE switch
VTSS_TARGET_SERVAL	Serval CE switch
VTSS_TARGET_SEVILLE	Seville switch
VTSS_TARGET_SPARX_III_10_UM	SparxIII-10 unmanaged switch
VTSS_TARGET_SPARX_III_17_UM	SparxIII-17 unmanaged switch
VTSS_TARGET_SPARX_III_25_UM	SparxIII-25 unmanaged switch
VTSS_TARGET_CARACAL_LITE	Caracal-Lite CE switch
VTSS_TARGET_SPARX_III_10	SparxIII-10 switch
VTSS_TARGET_SPARX_III_18	SparxIII-18 switch
VTSS_TARGET_SPARX_III_24	SparxIII-24 switch
VTSS_TARGET_SPARX_III_26	SparxIII-26 switch
VTSS_TARGET_SPARX_III_10_01	SparxIII-10-01 switch

## Enumerator

VTSS_TARGET_CARACAL_1	Caracal-1 CE switch
VTSS_TARGET_CARACAL_2	Caracal-2 CE switch
VTSS_TARGET_JAGUAR_1	Jaguar-1 CE switch
VTSS_TARGET_LYNX_1	LynX-1 CE switch
VTSS_TARGET_E_STAX_III_48	E-StaX-III-48
VTSS_TARGET_E_STAX_III_68	E-StaX-III-68
VTSS_TARGET_E_STAX_III_24_DUAL	Dual E-StaX-III-24
VTSS_TARGET_E_STAX_III_68_DUAL	Dual E-StaX-III-68
VTSS_TARGET_DAYTONA	Daytona FEC OTN Phy
VTSS_TARGET_TALLADEGA	Talladega FEC OTN Phy
VTSS_TARGET_SERVAL_2	Serval-2 CE switch
VTSS_TARGET_LYNX_2	LynX-2 CE switch
VTSS_TARGET_JAGUAR_2	Jaguar-2 CE switch
VTSS_TARGET_SPARX_IV_52	Sparx-IV-52 switch
VTSS_TARGET_SPARX_IV_44	Sparx-IV-44 switch
VTSS_TARGET_SPARX_IV_80	Sparx-IV-80 switch
VTSS_TARGET_SPARX_IV_90	Sparx-IV-80 switch

Definition at line 42 of file vtss\_init\_api.h.

## 7.15.4.2 vtss\_port\_mux\_mode\_t

```
enum vtss_port_mux_mode_t
```

Port mux configuration.

## Enumerator

VTSS_PORT_MUX_MODE_0	Ports muxed to Serdes blocks: 3xQSGMII, 1x2G5, 1xSGMII
VTSS_PORT_MUX_MODE_1	Ports muxed to Serdes blocks: 2x2G5, 10xSGMII
VTSS_PORT_MUX_MODE_2	Ports muxed to Serdes blocks: 2xQSGMII, 8xSGMII

Definition at line 335 of file vtss\_init\_api.h.

## 7.15.4.3 vtss\_restart\_t

```
enum vtss_restart_t
```

Restart type.

## Enumerator

VTSS_RESTART_COLD	Cold: Chip and CPU restart, e.g. power cycling
VTSS_RESTART_COOL	Cool: Chip and CPU restart done by CPU
VTSS_RESTART_WARM	Warm: CPU restart only

Definition at line 601 of file vtss\_init\_api.h.

## 7.15.5 Function Documentation

### 7.15.5.1 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.

### 7.15.5.2 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.

**7.15.5.3 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.

**7.15.5.4 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.

**7.15.5.5 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.

**7.15.5.6 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.

**7.15.5.7 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.

### 7.15.5.8 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.

### 7.15.5.9 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.

## 7.16 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\_LLC.*
- struct [vtss\\_vce\\_frame\\_snap\\_t](#)  
*Frame data for VTSS\_VCE\_TYPE\_SNAP.*
- struct [vtss\\_vce\\_frame\\_ipv4\\_t](#)  
*Frame data for VTSS\_VCE\_TYPE\_IPV4.*
- struct [vtss\\_vce\\_frame\\_ipv6\\_t](#)  
*Frame data for VTSS\_VCE\_TYPE\_IPV6.*
- struct [vtss\\_vce\\_key\\_t](#)  
*VCE Key.*
- struct [vtss\\_vce\\_action\\_t](#)  
*VCE Action.*
- struct [vtss\\_vce\\_t](#)  
*VLAN Control Entry.*
- struct [vtss\\_vlan\\_trans\\_port2grp\\_conf\\_t](#)  
*VLAN translation port-to-group configuration.*
- struct [vtss\\_vlan\\_trans\\_grp2vlan\\_conf\\_t](#)  
*VLAN translation group-to-VLAN configuration.*
- struct [vtss\\_dgroup\\_port\\_conf\\_t](#)  
*Destination group port configuration.*
- struct [vtss\\_sflow\\_port\\_conf\\_t](#)  
*sFlow configuration structure.*
- struct [vtss\\_mirror\\_conf\\_t](#)  
*Mirror configuration.*
- struct [vtss\\_eps\\_port\\_conf\\_t](#)  
*Port protection configuration.*

## Macros

- `#define VTSS_MAC_ADDRS 8192`
- `#define VTSS_MSTIS (65)`
- `#define VTSS_MSTI_START (0)`
- `#define VTSS_MSTI_END (VTSS_MSTI_START+VTSS_MSTIS)`
- `#define VTSS_MSTI_ARRAY_SIZE VTSS_MSTI_END`
- `#define VTSS_VCL_IDS 256`
- `#define VTSS_VCL_ID_START 0`
- `#define VTSS_VCL_ID_END (VTSS_VCL_ID_START+VTSS_VCL_IDS)`
- `#define VTSS_VCL_ARRAY_SIZE VTSS_VCL_ID_END`
- `#define VTSS_VCE_ID_LAST 0`
- `#define VTSS_VLAN_TRANS_GROUP_MAX_CNT VTSS_PORTS`
- `#define VTSS_VLAN_TRANS_MAX_CNT 256`
- `#define VTSS_VLAN_TRANS_NULL_GROUP_ID 0`
- `#define VTSS_VLAN_TRANS_FIRST_GROUP_ID 1`
- `#define VTSS_VLAN_TRANS_VID_START 1`
- `#define VTSS_VLAN_TRANS_MAX_VLAN_ID 4095`
- `#define VTSS_VLAN_TRANS_LAST_GROUP_ID (VTSS_VLAN_TRANS_FIRST_GROUP_ID + VTSS_VLAN_TRANS_GROUP - 1)`
- `#define VTSS_VLAN_TRANS_VALID_GROUP_CHECK(grp_id)`
- `#define VTSS_VLAN_TRANS_VALID_VLAN_CHECK(vid)`
- `#define VTSS_VLAN_TRANS_NULL_CHECK(ptr) ((ptr == NULL) ? FALSE : TRUE)`
- `#define VTSS_VLAN_TRANS_PORT_BF_SIZE ((VTSS_PORTS + 7)/8)`
- `#define VTSS_PVLANS (VTSS_PORTS)`
- `#define VTSS_PVLAN_NO_START (0)`
- `#define VTSS_PVLAN_NO_END (VTSS_PVLAN_NO_START+VTSS_PVLANS)`
- `#define VTSS_PVLAN_ARRAY_SIZE VTSS_PVLAN_NO_END`
- `#define VTSS_PVLAN_NO_DEFAULT (0)`
- `#define VTSS_ERPIS (64)`
- `#define VTSS_ERPI_START (0)`
- `#define VTSS_ERPI_END (VTSS_ERPI_START+VTSS_ERPIS)`
- `#define VTSS_ERPI_ARRAY_SIZE VTSS_ERPI_END`

## Typedefs

- `typedef u32 vtss_mac_table_age_time_t`  
*MAC address table age time.*
- `typedef u32 vtss_msti_t`  
*MSTP instance number.*
- `typedef u32 vtss_vce_id_t`  
*VCE ID type.*
- `typedef u64 vtss_vt_id_t`
- `typedef u32 vtss_pvlan_no_t`  
*Private VLAN Number.*
- `typedef vtss_port_no_t vtss_dgroup_no_t`  
*EVC policer configuration.*
- `typedef u32 vtss_erpi_t`  
*ERPS instance number.*

## Enumerations

- enum `vtss_stp_state_t` { `VTSS_STP_STATE_DISCARDING`, `VTSS_STP_STATE_LEARNING`, `VTSS_STP_STATE_FORWARDING` }

*Spanning Tree state.*

- enum `vtss_vlan_port_type_t` { `VTSS_VLAN_PORT_TYPE_UNAWARE`, `VTSS_VLAN_PORT_TYPE_C`, `VTSS_VLAN_PORT_TYPE_S`, `VTSS_VLAN_PORT_TYPE_S_CUSTOM` }

*VLAN port type configuration.*

- enum `vtss_vlan_tx_tag_t` { `VTSS_VLAN_TX_TAG_PORT`, `VTSS_VLAN_TX_TAG_DISABLE`, `VTSS_VLAN_TX_TAG_ENABLE` }

*VLAN Tx tag type.*

- enum `vtss_vce_type_t` { `VTSS_VCE_TYPE_ANY`, `VTSS_VCE_TYPEETYPE`, `VTSS_VCE_TYPE_LLCC`, `VTSS_VCE_TYPE_SNAP`, `VTSS_VCE_TYPE_IPV4`, `VTSS_VCE_TYPE_IPV6` }

*VCE frame type.*

- 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.*

## 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_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)

- `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)
  - Get 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)
  - Set Destination Group configuration for port.*
- `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)
  - Get 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_mirror_conf_get` (const `vtss_inst_t` inst, `vtss_mirror_conf_t` \*const conf)
  - Get the mirror configuration.*
- `vtss_rc vtss_mirror_conf_set` (const `vtss_inst_t` inst, const `vtss_mirror_conf_t` \*const conf)
  - Set the mirror configuration.*
- `vtss_rc vtss_mirror_monitor_port_get` (const `vtss_inst_t` inst, `vtss_port_no_t` \*const port\_no)
  - Get the mirror monitor port.*
- `vtss_rc vtss_mirror_monitor_port_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no)
  - Set the mirror monitor port.*
- `vtss_rc vtss_mirror_ingress_ports_get` (const `vtss_inst_t` inst, `BOOL` member[`VTSS_PORT_ARRAY_SIZE`])
  - Get the mirror ingress ports.*
- `vtss_rc vtss_mirror_ingress_ports_set` (const `vtss_inst_t` inst, const `BOOL` member[`VTSS_PORT_ARRAY_SIZE`])
  - Set the mirror ingress ports.*
- `vtss_rc vtss_mirror_egress_ports_get` (const `vtss_inst_t` inst, `BOOL` member[`VTSS_PORT_ARRAY_SIZE`])
  - Get the mirror egress ports.*
- `vtss_rc vtss_mirror_egress_ports_set` (const `vtss_inst_t` inst, const `BOOL` member[`VTSS_PORT_ARRAY_SIZE`])
  - Set the mirror egress ports.*
- `vtss_rc vtss_mirror_cpu_ingress_get` (const `vtss_inst_t` inst, `BOOL` \*member)
  - Get the mirror CPU ingress.*
- `vtss_rc vtss_mirror_cpu_ingress_set` (const `vtss_inst_t` inst, const `BOOL` member)
  - Set CPU ingress mirroring.*
- `vtss_rc vtss_mirror_cpu_egress_get` (const `vtss_inst_t` inst, `BOOL` \*member)
  - Get the mirror CPU egress.*
- `vtss_rc vtss_mirror_cpu_egress_set` (const `vtss_inst_t` inst, const `BOOL` member)
  - Set the mirror CPU egress.*
- `vtss_rc vtss_uc_flood_members_get` (const `vtss_inst_t` inst, `BOOL` member[`VTSS_PORT_ARRAY_SIZE`])
  - Get unicast flood members.*
- `vtss_rc vtss_uc_flood_members_set` (const `vtss_inst_t` inst, const `BOOL` member[`VTSS_PORT_ARRAY_SIZE`])
  - Set unicast flood members.*

- `vtss_rc vtss_mc_flood_members_get (const vtss_inst_t inst, BOOL member[VTSS_PORT_ARRAY_SIZE])`  
*Get multicast flood members.*
- `vtss_rc vtss_mc_flood_members_set (const vtss_inst_t inst, const BOOL member[VTSS_PORT_ARRAY_SIZE])`  
*Set multicast flood members.*
- `vtss_rc vtss_ipv4_mc_flood_members_get (const vtss_inst_t inst, BOOL member[VTSS_PORT_ARRAY_SIZE])`  
*Get IPv4 multicast flood members.*
- `vtss_rc vtss_ipv4_mc_flood_members_set (const vtss_inst_t inst, const BOOL member[VTSS_PORT_ARRAY_SIZE])`  
*Set IPv4 multicast flood members.*
- `vtss_rc vtss_ipv4_mc_add (const vtss_inst_t inst, const vtss_vid_t vid, const vtss_ip_t sip, const vtss_ip_t dip, const BOOL member[VTSS_PORT_ARRAY_SIZE])`  
*Add IPv4 multicast entry.*
- `vtss_rc vtss_ipv4_mc_del (const vtss_inst_t inst, const vtss_vid_t vid, const vtss_ip_t sip, const vtss_ip_t dip)`  
*Delete IPv4 multicast entry.*
- `vtss_rc vtss_ipv6_mc_flood_members_get (const vtss_inst_t inst, BOOL member[VTSS_PORT_ARRAY_SIZE])`  
*Get IPv6 multicast flood members.*
- `vtss_rc vtss_ipv6_mc_flood_members_set (const vtss_inst_t inst, const BOOL member[VTSS_PORT_ARRAY_SIZE])`  
*Set IPv6 multicast flood members.*
- `vtss_rc vtss_ipv6_mc_ctrl_flood_get (const vtss_inst_t inst, BOOL *const scope)`  
*Get IPv6 multicast control flooding mode.*
- `vtss_rc vtss_ipv6_mc_ctrl_flood_set (const vtss_inst_t inst, const BOOL scope)`  
*Set IPv6 multicast control flooding mode. This controls whether unknown Link-Local scope IPv6 multicasts (FF02::/16) are flooded to all ports or to the ports in the IPv6 multicast flood mask.*
- `vtss_rc vtss_ipv6_mc_add (const vtss_inst_t inst, const vtss_vid_t vid, const vtss_ipv6_t sip, const vtss_ipv6_t dip, const BOOL member[VTSS_PORT_ARRAY_SIZE])`  
*Add IPv6 multicast entry.*
- `vtss_rc vtss_ipv6_mc_del (const vtss_inst_t inst, const vtss_vid_t vid, const vtss_ipv6_t sip, const vtss_ipv6_t dip)`  
*Delete IPv6 multicast entry.*
- `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.*

### 7.16.1 Detailed Description

Layer 2 API.

This header file describes Layer 2 switching functions

### 7.16.2 Macro Definition Documentation

#### 7.16.2.1 VTSS\_MAC\_ADDRS

```
#define VTSS_MAC_ADDRS 8192
```

Number of MAC addresses

Definition at line 95 of file vtss\_l2\_api.h.

#### 7.16.2.2 VTSS\_MSTIS

```
#define VTSS_MSTIS (65)
```

Number of MSTP instances

Definition at line 514 of file vtss\_l2\_api.h.

#### 7.16.2.3 VTSS\_MSTI\_START

```
#define VTSS_MSTI_START (0)
```

MSTI start number

Definition at line 515 of file vtss\_l2\_api.h.

#### 7.16.2.4 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.

### 7.16.2.5 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.

### 7.16.2.6 VTSS\_VCL\_IDS

```
#define VTSS_VCL_IDS 256
```

Number of VCLs

Definition at line 922 of file vtss\_l2\_api.h.

### 7.16.2.7 VTSS\_VCL\_ID\_START

```
#define VTSS_VCL_ID_START 0
```

VCL ID start number

Definition at line 923 of file vtss\_l2\_api.h.

### 7.16.2.8 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.

### 7.16.2.9 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.

### 7.16.2.10 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.

### 7.16.2.11 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.

### 7.16.2.12 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.

### 7.16.2.13 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.

### 7.16.2.14 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.

### 7.16.2.15 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.

### 7.16.2.16 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.

### 7.16.2.17 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.

### 7.16.2.18 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.

### 7.16.2.19 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.

### 7.16.2.20 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.

### 7.16.2.21 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.

### 7.16.2.22 VTSS\_PVLANS

```
#define VTSS_PVLANS (VTSS_PORTS)
```

Number of PVLANS

Definition at line 1304 of file vtss\_l2\_api.h.

### 7.16.2.23 VTSS\_PVLAN\_NO\_START

```
#define VTSS_PVLAN_NO_START (0)
```

PVLAN start number

Definition at line 1305 of file vtss\_l2\_api.h.

### 7.16.2.24 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.

### 7.16.2.25 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.

### 7.16.2.26 VTSS\_PVLAN\_NO\_DEFAULT

```
#define VTSS_PVLAN_NO_DEFAULT (0)
```

Default PVLAN

Definition at line 1308 of file vtss\_l2\_api.h.

### 7.16.2.27 VTSS\_ERPIS

```
#define VTSS_ERPIS (64)
```

Number of ERPS instances

Definition at line 2228 of file vtss\_l2\_api.h.

### 7.16.2.28 VTSS\_ERPI\_START

```
#define VTSS_ERPI_START (0)
```

ERPI start number

Definition at line 2229 of file vtss\_l2\_api.h.

### 7.16.2.29 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.

### 7.16.2.30 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.

## 7.16.3 Typedef Documentation

### 7.16.3.1 vtss\_vt\_id\_t

```
typedef u64 vtss_vt_id_t
```

VLAN Translation ID

Definition at line 1095 of file vtss\_l2\_api.h.

## 7.16.4 Enumeration Type Documentation

### 7.16.4.1 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.

## 7.16.4.2 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.

## 7.16.4.3 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.

## 7.16.4.4 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.

**7.16.4.5 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.

**7.16.4.6 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.

**7.16.4.7 vtss\_erps\_state\_t**

```
enum vtss_erps_state_t
```

ERPS state.

## Enumerator

VTSS_ERPS_STATE_FORWARDING	Forwarding
VTSS_ERPS_STATE_DISCARDING	Discardng

Definition at line 2266 of file vtss\_l2\_api.h.

## 7.16.5 Function Documentation

### 7.16.5.1 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.

### 7.16.5.2 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.

### 7.16.5.3 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.

### 7.16.5.4 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.

### 7.16.5.5 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.

**7.16.5.6 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.

**7.16.5.7 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.

### 7.16.5.8 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.

### 7.16.5.9 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.

### 7.16.5.10 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.

**7.16.5.11 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.

**7.16.5.12 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.

**7.16.5.13 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.

### 7.16.5.14 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.

### 7.16.5.15 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.

**7.16.5.16 vtss\_port\_state\_get()**

```
vtss_rc vtss_port_state_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    BOOL *const state )
```

Get port operational state.

**Parameters**

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>state</i>	[OUT] Port state, TRUE if link is up.

**Returns**

Return code.

**7.16.5.17 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.

### 7.16.5.18 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.

### 7.16.5.19 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.

### 7.16.5.20 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.

**7.16.5.21 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.

**7.16.5.22 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.

**7.16.5.23 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.

**7.16.5.24 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.

### 7.16.5.25 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.

### 7.16.5.26 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.

### 7.16.5.27 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.

**7.16.5.28 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.

**7.16.5.29 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.

**7.16.5.30 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.

**7.16.5.31 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.

### 7.16.5.32 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.

### 7.16.5.33 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.

### 7.16.5.34 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.

**7.16.5.35 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 )
```

Get 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.

**7.16.5.36 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.

**7.16.5.37 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.

**7.16.5.38 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.

## 7.16.5.39 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.

## 7.16.5.40 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.

## 7.16.5.41 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.

### 7.16.5.42 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.

### 7.16.5.43 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.

**7.16.5.44 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.

**7.16.5.45 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.

#### 7.16.5.46 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.

#### 7.16.5.47 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.

#### 7.16.5.48 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.

**7.16.5.49 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.

**7.16.5.50 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.

**7.16.5.51 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.

**7.16.5.52 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.

### 7.16.5.53 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.

### 7.16.5.54 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.

### 7.16.5.55 vtss\_sflow\_port\_conf\_set()

```
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.

**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.

**7.16.5.56 vtss\_sfloop\_sampling\_rate\_convert()**

```
vtss_rc vtss_sfloop_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.

**7.16.5.57 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.

**7.16.5.58 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.

**7.16.5.59 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.

**7.16.5.60 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.

**7.16.5.61 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.

**7.16.5.62 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.

**7.16.5.63 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.

**7.16.5.64 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.

### 7.16.5.65 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.

### 7.16.5.66 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.

### 7.16.5.67 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.

**7.16.5.68 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.

**7.16.5.69 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.

**7.16.5.70 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.

### 7.16.5.71 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.

### 7.16.5.72 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.

**7.16.5.73 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.

**7.16.5.74 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.

**7.16.5.75 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.

**7.16.5.76 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.

**7.16.5.77 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.

## 7.16.5.78 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.

## 7.16.5.79 vtss\_ipv4\_mc\_add()

```
vtss_rc vtss_ipv4_mc_add (
    const vtss_inst_t inst,
    const vtss_vid_t vid,
    const vtss_ip_t sip,
    const vtss_ip_t dip,
    const BOOL member[VTSS_PORT_ARRAY_SIZE] )
```

Add IPv4 multicast entry.

## Parameters

<i>inst</i>	[IN] Target instance reference.
<i>vid</i>	[IN] VLAN ID.
<i>sip</i>	[IN] Source IP address.
<i>dip</i>	[IN] Destination IP address.
<i>member</i>	[IN] Port member list.

## Returns

Return code.

## 7.16.5.80 vtss\_ipv4\_mc\_del()

```
vtss_rc vtss_ipv4_mc_del (
    const vtss_inst_t inst,
    const vtss_vid_t vid,
```

```
const vtss_ip_t sip,  
const vtss_ip_t dip )
```

Delete IPv4 multicast entry.

**Parameters**

<i>inst</i>	[IN] Target instance reference.
<i>vid</i>	[IN] VLAN ID.
<i>sip</i>	[IN] Source IP address.
<i>dip</i>	[IN] Destination IP address.

**Returns**

Return code.

**7.16.5.81 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.

**7.16.5.82 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.

### 7.16.5.83 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.

### 7.16.5.84 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.

### 7.16.5.85 vtss\_ipv6\_mc\_add()

```
vtss_rc vtss_ipv6_mc_add (
    const vtss_inst_t inst,
    const vtss_vid_t vid,
    const vtss_ipv6_t sip,
    const vtss_ipv6_t dip,
    const BOOL member[VTSS_PORT_ARRAY_SIZE] )
```

Add IPv6 multicast entry.

**Parameters**

<i>inst</i>	[IN] Target instance reference.
<i>vid</i>	[IN] VLAN ID.
<i>sip</i>	[IN] Source IP address.
<i>dip</i>	[IN] Destination IP address.
<i>member</i>	[IN] Port member list.

**Returns**

Return code.

**7.16.5.86 vtss\_ipv6\_mc\_del()**

```
vtss_rc vtss_ipv6_mc_del (
    const vtss_inst_t inst,
    const vtss_vid_t vid,
    const vtss_ipv6_t sip,
    const vtss_ipv6_t dip )
```

Delete IPv6 multicast entry.

**Parameters**

<i>inst</i>	[IN] Target instance reference.
<i>vid</i>	[IN] VLAN ID.
<i>sip</i>	[IN] Source IP address.
<i>dip</i>	[IN] Destination IP address.

**Returns**

Return code.

**7.16.5.87 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.

Generated by Doxygen

**Returns**

Return code.

**7.16.5.88 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.

**7.16.5.89 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.

## 7.16.5.90 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.

## 7.16.5.91 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.

## 7.16.5.92 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.

**7.16.5.93 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.

**7.16.5.94 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.

Generated by Doxygen

**Returns**

Return code.

## 7.17 vtss\_api/include/vtss\_l3\_api.h File Reference

L3 routing API.

```
#include <vtss/api/types.h>
```

### 7.17.1 Detailed Description

L3 routing API.

This header file describes L3 IPv4/IPv6 hardware assisted routing functions.

## 7.18 vtss\_api/include/vtss\_mac10g\_api.h File Reference

MAC10G API.

```
#include <vtss/api/types.h>
```

### 7.18.1 Detailed Description

MAC10G API.

## 7.19 vtss\_api/include/vtss\_misc\_api.h File Reference

Miscellaneous API.

```
#include <vtss/api/types.h>
#include "vtss_init_api.h"
```

## Data Structures

- struct `vtss_trace_conf_t`  
*Trace group configuration.*
- struct `vtss_debug_info_t`  
*Debug information structure.*
- struct `vtss_api_lock_t`  
*API lock structure.*
- struct `vtss_debug_lock_t`  
*API debug lock structure.*
- struct `vtss_chip_id_t`  
*Chip ID.*
- struct `vtss_sgpio_port_conf_t`  
*SGPIO port configuration.*
- struct `vtss_sgpio_conf_t`  
*SGPIO configuration for a group.*
- struct `vtss_sgpio_port_data_t`  
*SGPIO read data for a port.*
- struct `vtss_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` 32  
*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` 1  
*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_LR, 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` }  
*GPIO configured mode.*
- enum `vtss_sgpi_mode_t`{  
`VTSS_SGPI_MODE_OFF`, `VTSS_SGPI_MODE_ON`, `VTSS_SGPI_MODE_0`, `VTSS_SGPI_MODE_1`,  
`VTSS_SGPI_MODE_0_ACTIVITY`, `VTSS_SGPI_MODE_1_ACTIVITY`, `VTSS_SGPI_MODE_0_ACTIVITY_INV`,  
`VTSS_SGPI_MODE_1_ACTIVITY_INV` }  
*SGPIO output mode.*
- enum `vtss_sgpi_bmode_t`{  
`VTSS_SGPI_BMODE_TOGGLE`,     `VTSS_SGPI_BMODE_0_625`,     `VTSS_SGPI_BMODE_1_25`,  
`VTSS_SGPI_BMODE_2_5`,  
`VTSS_SGPI_BMODE_5` }  
*SGPIO blink mode.*
- enum `vtss_irq_t`{  
`VTSS_IRQ_XTR`, `VTSS_IRQ_FDMA_XTR`, `VTSS_IRQ_SOFTWARE`, `VTSS_IRQ_PTP_RDY`,  
`VTSS_IRQ_PTP_SYNC`, `VTSS_IRQ_EXT1`, `VTSS_IRQ_OAM`, `VTSS_IRQ_MAX` }  
*Interrupt sources.*
- enum `vtss_fan_pwd_freq_t`{  
`VTSS_FAN_PWM_FREQ_25KHZ`, `VTSS_FAN_PWM_FREQ_120HZ`, `VTSS_FAN_PWM_FREQ_100HZ`,  
`VTSS_FAN_PWM_FREQ_80HZ`,  
`VTSS_FAN_PWM_FREQ_60HZ`, `VTSS_FAN_PWM_FREQ_40HZ`, `VTSS_FAN_PWM_FREQ_20HZ`, `VTSS_FAN_PWM_FREQ_10HZ` }  
*FAN PWM frequency.*
- enum `vtss_fan_type_t`{ `VTSS_FAN_2_WIRE_TYPE`, `VTSS_FAN_3_WIRE_TYPE`, `VTSS_FAN_4_WIRE_TYPE` }  
*FAN Types.*
- enum `vtss_eee_state_select_t`{  
`VTSS_EEE_STATE_SELECT_LPI`, `VTSS_EEE_STATE_SELECT_SCH`, `VTSS_EEE_STATE_SELECT_FP`,  
`VTSS_EEE_STATE_SELECT_INTR_ENA`,  
`VTSS_EEE_STATE_SELECT_INTR_ACK` }  
*EEE port state change what? (JR only)*

## Functions

- `vtss_rc vtss_trace_conf_get (const vtss_trace_group_t group, vtss_trace_conf_t *const conf)`  
*Get trace configuration.*
- `vtss_rc vtss_trace_conf_set (const vtss_trace_group_t group, const vtss_trace_conf_t *const conf)`  
*Set trace configuration.*
- `void vtss_callout_trace_printf (const vtss_trace_layer_t layer, const vtss_trace_group_t group, const vtss_trace_level_t level, const char *file, const int line, const char *function, const char *format,...)`  
*Trace callout function.*

- void `vtss_callout_trace_hex_dump` (const `vtss_trace_layer_t` layer, const `vtss_trace_group_t` group, const `vtss_trace_level_t` level, const char \*file, const int line, const char \*function, const `u8` \*byte\_p, const int byte\_cnt)
 

*Trace hex-dump callout function.*
- `vtss_rc vtss_debug_info_get` (`vtss_debug_info_t` \*const info)
 

*Get default debug information structure.*
- `vtss_rc vtss_debug_info_print` (const `vtss_inst_t` inst, const `vtss_debug_printf_t` prnft, const `vtss_debug_info_t` \*const info)
 

*Print default information.*
- void `vtss_callout_lock` (const `vtss_api_lock_t` \*const lock)
 

*Lock API access.*
- void `vtss_callout_unlock` (const `vtss_api_lock_t` \*const lock)
 

*Unlock API access.*
- `vtss_rc vtss_debug_lock` (const `vtss_inst_t` inst, const `vtss_debug_lock_t` \*const lock)
 

*Debug lock API access.*
- `vtss_rc vtss_debug_unlock` (const `vtss_inst_t` inst, `vtss_debug_lock_t` \*const lock)
 

*Debug unlock API access.*
- `vtss_rc vtss_reg_read` (const `vtss_inst_t` inst, const `vtss_chip_no_t` chip\_no, const `u32` addr, `u32` \*const value)
 

*Read value from target register.*
- `vtss_rc vtss_reg_write` (const `vtss_inst_t` inst, const `vtss_chip_no_t` chip\_no, const `u32` addr, const `u32` value)
 

*Write value to target register.*
- `vtss_rc vtss_reg_write_masked` (const `vtss_inst_t` inst, const `vtss_chip_no_t` chip\_no, const `u32` addr, const `u32` value, const `u32` mask)
 

*Read, modify and write value to target register.*
- `vtss_rc vtss_intr_sticky_clear` (const `vtss_inst_t` inst, `u32` ext)
 

*Clear EXT0-1 interrupt sticky bits on secondary chip.*
- `vtss_rc vtss_chip_id_get` (const `vtss_inst_t` inst, `vtss_chip_id_t` \*const chip\_id)
 

*Get chip ID and revision.*
- `vtss_rc vtss_poll_1sec` (const `vtss_inst_t` inst)
 

*Polling function called every second.*
- `vtss_rc vtss_ptp_event_poll` (const `vtss_inst_t` inst, `vtss_ptp_event_type_t` \*const ev\_mask)
 

*PTP polling function called at by interrupt or periodically.*
- `vtss_rc vtss_ptp_event_enable` (const `vtss_inst_t` inst, const `vtss_ptp_event_type_t` ev\_mask, const `BOOL` enable)
 

*Enable PTP event generation for a specific event type.*
- `vtss_rc vtss_dev_all_event_poll` (const `vtss_inst_t` inst, const `vtss_dev_all_event_poll_t` poll\_type, `vtss_dev_all_event_type_t` \*const ev\_mask)
 

*DEV\_ALL polling function called at by interrupt or periodically.*
- `vtss_rc vtss_dev_all_event_enable` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_dev_all_event_type_t` ev\_mask, const `BOOL` enable)
 

*Enable DEV\_ALL event generation for a specific event type.*
- `vtss_rc vtss_gpio_mode_set` (const `vtss_inst_t` inst, const `vtss_chip_no_t` chip\_no, const `vtss_gpio_no_t` gpio\_no, const `vtss_gpio_mode_t` mode)
 

*Set GPIO mode.*
- `vtss_rc vtss_gpio_direction_set` (const `vtss_inst_t` inst, const `vtss_chip_no_t` chip\_no, const `vtss_gpio_no_t` gpio\_no, const `BOOL` output)
 

*Set GPIO direction to input or output.*
- `vtss_rc vtss_gpio_read` (const `vtss_inst_t` inst, const `vtss_chip_no_t` chip\_no, const `vtss_gpio_no_t` gpio\_no, `BOOL` \*const value)
 

*Read from GPIO input pin.*

- `vtss_rc vtss_gpio_write` (const `vtss_inst_t` inst, const `vtss_chip_no_t` chip\_no, const `vtss_gpio_no_t` gpio\_no, const `BOOL` value)
 

*Write to GPIO output pin.*
- `vtss_rc vtss_gpio_event_poll` (const `vtss_inst_t` inst, const `vtss_chip_no_t` chip\_no, `BOOL` \*const events)
 

*Get GPIO event indication.*
- `vtss_rc vtss_gpio_event_enable` (const `vtss_inst_t` inst, const `vtss_chip_no_t` chip\_no, const `vtss_gpio_no_t` gpio\_no, `BOOL` enable)
 

*Set GPIO event enable.*
- `vtss_rc vtss_sgpi_conf_get` (const `vtss_inst_t` inst, const `vtss_chip_no_t` chip\_no, const `vtss_sgpi_group_t` group, `vtss_sgpi_conf_t` \*const conf)
 

*Get SGPIO configuration.*
- `vtss_rc vtss_sgpi_conf_set` (const `vtss_inst_t` inst, const `vtss_chip_no_t` chip\_no, const `vtss_sgpi_group_t` group, const `vtss_sgpi_conf_t` \*const conf)
 

*Set SGPIO configuration.*
- `vtss_rc vtss_sgpi_read` (const `vtss_inst_t` inst, const `vtss_chip_no_t` chip\_no, const `vtss_sgpi_group_t` group, `vtss_sgpi_port_data_t` data[`VTSS_SGPIO_PORTS`])
 

*Read SGPIO data.*
- `vtss_rc vtss_sgpi_event_poll` (const `vtss_inst_t` inst, const `vtss_chip_no_t` chip\_no, const `vtss_sgpi_group_t` group, const `u32` bit, `BOOL` \*const events)
 

*Get SGPIO event indication.*
- `vtss_rc vtss_sgpi_event_enable` (const `vtss_inst_t` inst, const `vtss_chip_no_t` chip\_no, const `vtss_sgpi_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.*

### 7.19.1 Detailed Description

Miscellaneous API.

This header file describes miscellaneous API functions

### 7.19.2 Macro Definition Documentation

#### 7.19.2.1 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.

#### 7.19.2.2 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.

### 7.19.3 Typedef Documentation

#### 7.19.3.1 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.

### 7.19.4 Enumeration Type Documentation

#### 7.19.4.1 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.

#### 7.19.4.2 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

## Enumerator

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.

## 7.19.4.3 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.

## 7.19.4.4 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.

## 7.19.4.5 vtss\_debug\_group\_t

```
enum vtss_debug_group_t
```

Debug function group.

## Enumerator

VTSS_DEBUG_GROUP_ALL	All groups
VTSS_DEBUG_GROUP_INIT	Initialization
VTSS_DEBUG_GROUP_MISC	Miscellaneous
VTSS_DEBUG_GROUP_PORT	Port configuration
VTSS_DEBUG_GROUP_PORT_CNT	Port counters
VTSS_DEBUG_GROUP_PHY	PHY
VTSS_DEBUG_GROUP_VLAN	VLAN
VTSS_DEBUG_GROUP_PVLAN	PVLAN
VTSS_DEBUG_GROUP_MAC_TABLE	MAC address table
VTSS_DEBUG_GROUP_ACL	ACL
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

## Enumerator

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.

## 7.19.4.6 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.

## 7.19.4.7 vtss\_sgpi\_mode\_t

```
enum vtss_sgpi_mode_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.

#### 7.19.4.8 vtss\_sgpiobmode\_t

enum `vtss_sgpiobmode_t`

SGPIO blink mode.

**Enumerator**

<code>VTSS_SGPIO_BMODE_TOGGLE</code>	Burst toggle (mode 1 only)
<code>VTSS_SGPIO_BMODE_0_625</code>	0.625 Hz (mode 0 only)
<code>VTSS_SGPIO_BMODE_1_25</code>	1.25 Hz
<code>VTSS_SGPIO_BMODE_2_5</code>	2.5 Hz
<code>VTSS_SGPIO_BMODE_5</code>	5 Hz

Definition at line 779 of file vtss\_misc\_api.h.

#### 7.19.4.9 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.

#### 7.19.4.10 vtss\_eee\_state\_select\_t

enum `vtss_eee_state_select_t`

EEE port state change what? (JR only)

## Enumerator

VTSS_EEE_STATE_SELECT_LPI	Change LPI signal.
VTSS_EEE_STATE_SELECT_SCH	Change scheduler enable.
VTSS_EEE_STATE_SELECT_FP	Change frame mirroring flag.
VTSS_EEE_STATE_SELECT_INTR_ENA	Enable analyzer interrupts.
VTSS_EEE_STATE_SELECT_INTR_ACK	Acknowledge analyzer interrupts.

Definition at line 1230 of file vtss\_misc\_api.h.

### 7.19.5 Function Documentation

#### 7.19.5.1 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.

#### 7.19.5.2 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.

**7.19.5.3 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.

**7.19.5.4 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.

**7.19.5.5 vtss\_debug\_info\_get()**

```
vtss_rc vtss_debug_info_get (
    vtss_debug_info_t *const info )
```

Get default debug information structure.

**Parameters**

<i>info</i>	[OUT] Debug information
-------------	-------------------------

**Returns**

Return code.

**7.19.5.6 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.

**7.19.5.7 vtss\_callout\_lock()**

```
void vtss_callout_lock (
    const vtss_api_lock_t *const lock )
```

Lock API access.

**Parameters**

<i>lock</i>	[IN] Lock information
-------------	-----------------------

**7.19.5.8 vtss\_callout\_unlock()**

```
void vtss_callout_unlock (
    const vtss_api_lock_t *const lock )
```

Unlock API access.

**Parameters**

<i>lock</i>	[IN] Lock information
-------------	-----------------------

**7.19.5.9 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.

### 7.19.5.10 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.

### 7.19.5.11 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.

### 7.19.5.12 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.

### 7.19.5.13 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.

### 7.19.5.14 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.

**7.19.5.15 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.

**7.19.5.16 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.

### 7.19.5.17 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.

### 7.19.5.18 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.

### 7.19.5.19 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.

### 7.19.5.20 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.

### 7.19.5.21 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.

### 7.19.5.22 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.

### 7.19.5.23 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.

**7.19.5.24 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.

**7.19.5.25 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.

**7.19.5.26 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.

**7.19.5.27 vtss\_sgpi\_conf\_get()**

```
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.

**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>conf</i>	[OUT] SGPIO configuration.

**Returns**

Return code.

### 7.19.5.28 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.

### 7.19.5.29 vtss\_sgpio\_read()

```
vtss_rc vtss_sgpio_read (
    const vtss_inst_t inst,
    const vtss_chip_no_t chip_no,
    const vtss_sgpio_group_t group,
    vtss_sgpio_port_data_t data[VTSS_SGPIO_PORTS] )
```

Read GPIO data.

#### Parameters

<i>inst</i>	[IN] Target instance reference.
<i>chip_no</i>	[IN] Chip number (if multi-chip instance).
<i>group</i>	[IN] GPIO group.
<i>data</i>	[OUT] GPIO data.

#### Returns

Return code.

### 7.19.5.30 vtss\_sgpio\_event\_poll()

```
vtss_rc vtss_sgpio_event_poll (
    const vtss_inst_t inst,
```

```
const vtss_chip_no_t chip_no,
const vtss_sgpio_group_t group,
const u32 bit,
BOOL *const events )
```

Get GPIO event indication.

#### Parameters

<i>inst</i>	[IN] Target instance reference.
<i>chip_no</i>	[IN] Chip number (if multi-chip instance).
<i>group</i>	[IN] GPIO group.
<i>bit</i>	[IN] GPIO port bit (0-3).
<i>events</i>	[OUT] Event indication for each port for the selected bit - must point to VTSS_SGPIO_PORTS of BOOL.

#### Returns

Return code.

### 7.19.5.31 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.

### 7.19.5.32 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.

### 7.19.5.33 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.

### 7.19.5.34 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.

**7.19.5.35 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.

**7.19.5.36 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.

**7.19.5.37 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.

**7.19.5.38 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.

**7.19.5.39 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.

**7.19.5.40 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

**7.19.5.41 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
-----------	------------------------------

**7.19.5.42 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.

**7.19.5.43 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.

**7.19.5.44 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.

#### 7.19.5.45 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.

#### 7.19.5.46 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.

#### 7.19.5.47 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.

**7.19.5.48 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.

**7.19.5.49 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.

**7.19.5.50 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.

**Returns**

Return code.

**7.19.5.51 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.

## 7.20 vtss\_api/include/vtss\_mpls\_api.h File Reference

MPLS API.

```
#include <vtss/api/types.h>
```

### 7.20.1 Detailed Description

MPLS API.

This header file describes the MPLS functions

## 7.21 vtss\_api/include/vtss\_oam\_api.h File Reference

OAM API.

```
#include <vtss/api/types.h>
```

### 7.21.1 Detailed Description

OAM API.

This header file describes Y.1731/IEEE802.1ag OAM functions.

## 7.22 vtss\_api/include/vtss\_oha\_api.h File Reference

OHA API.

```
#include <vtss/api/types.h>
```

### 7.22.1 Detailed Description

OHA API.

## 7.23 vtss\_api/include/vtss\_os.h File Reference

OS Layer API.

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

### 7.23.1 Detailed Description

OS Layer API.

This header file includes the OS specific header file

## 7.24 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`

### 7.24.1 Detailed Description

OS custom header file.

This file is a skeleton to be replaced by for customer specific OS.

### 7.24.2 Macro Definition Documentation

#### 7.24.2.1 uint

```
#define uint unsigned int
```

Define API uint type - unsigned 16 bits

Definition at line 36 of file vtss\_os\_custom.h.

#### 7.24.2.2 ulong

```
#define ulong unsigned long
```

Define API ulong - unsigned 32 bits

Definition at line 37 of file vtss\_os\_custom.h.

#### 7.24.2.3 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.

#### 7.24.2.4 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.

#### 7.24.2.5 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.

#### 7.24.2.6 VTSS\_MTIMER\_CANCEL

```
#define VTSS_MTIMER_CANCEL(  
    pTimer ) <your impl>
```

Stop the timer

Definition at line 46 of file vtss\_os\_custom.h.

#### 7.24.2.7 VTSS\_DIV64

```
#define VTSS_DIV64(  
    dividend,  
    divisor ) <your impl>
```

support for 64 bit division

Definition at line 48 of file vtss\_os\_custom.h.

#### 7.24.2.8 VTSS\_MOD64

```
#define VTSS_MOD64(  
    dividend,  
    divisor ) <your impl>
```

support for 64 bit division

Definition at line 49 of file vtss\_os\_custom.h.

### 7.24.2.9 VTSS\_LABS

```
#define VTSS_LABS(  
    arg ) <your impl>
```

long to abs

Definition at line 50 of file vtss\_os\_custom.h.

### 7.24.2.10 VTSS\_LLABS

```
#define VTSS_LLABS(  
    arg ) <your impl>
```

long long to abs

Definition at line 51 of file vtss\_os\_custom.h.

### 7.24.2.11 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.

### 7.24.2.12 VTSS\_OS\_CTZ64

```
#define VTSS_OS_CTZ64(  
    val64 ) <your impl>
```

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 73 of file vtss\_os\_custom.h.

### 7.24.2.13 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.

### 7.24.2.14 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.

### 7.24.2.15 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.

## 7.24.3 Typedef Documentation

### 7.24.3.1 vtss\_mtimer\_t

```
typedef int vtss_mtimer_t
```

Timer

Definition at line 43 of file vtss\_os\_custom.h.

## 7.25 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_Ctz`(val32) `((val32) == 0 ? 32 : __builtin_ctz(val32))`
- #define `VTSS_OS_Ctz64`(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_REORDER_BARRIER`() `HAL_REORDER_BARRIER()`
- #define `VTSS_OS_COMPILER_ATTRIBUTE_ALIGNED`(x) `__attribute__((aligned(x)))`
- #define `VTSS_OS_DCACHE_LINE_SIZE_BYTES` `HAL_DCACHE_LINE_SIZE`
- #define `VTSS_OS_DCACHE_INVALIDATE`(virt\_addr, size) `HAL_DCACHE_INVALIDATE(virt_addr, size)`
- #define `VTSS_OS_DCACHE_FLUSH`(virt\_addr, size) `HAL_DCACHE_STORE(virt_addr, size)`
- #define `VTSS_OS_VIRT_TO_PHYS`(addr) `(u32)CYGARC_PHYSICAL_ADDRESS(addr)`
- #define `VTSS_OS_BIG_ENDIAN`

*VTSS\_OS\_BIG\_ENDIAN: If undefined, we're running little endian. If defined we're running big endian.*

  - #define `VTSS_OS_NTOHL`(x) `(x)`
  - #define `VTSS_OS_SCHEDULER_FLAGS` `cyg_uint32 __attribute__((unused))`
  - #define `VTSS_OS_SCHEDULER_LOCK`(flags) `cyg_scheduler_lock(__FILE__, __LINE__)`
  - #define `VTSS_OS_SCHEDULER_UNLOCK`(flags) `cyg_scheduler_unlock(__FILE__, __LINE__)`
  - #define `VTSS_OS_INTERRUPT_FLAGS` `NOT_NEEDED`
  - #define `VTSS_OS_INTERRUPT_DISABLE`(flags) `NOT_NEEDED`
  - #define `VTSS_OS_INTERRUPT_RESTORE`(flags) `NOT_NEEDED`

## Typedefs

- `typedef cyg_tick_count_t vtss_mtimer_t`

## Functions

- `long long int llabs (long long int val)`  
*Obtain the absolute value of a long long integer.*
- `void * vtss_callout_malloc (size_t size, vtss_mem_flags_t flags)`  
*Callout to allocate memory.*
- `void vtss_callout_free (void *ptr, vtss_mem_flags_t flags)`  
*Callout to free memory.*

### 7.25.1 Detailed Description

#### eCos OS API

This header file describes OS functions for eCos

### 7.25.2 Macro Definition Documentation

#### 7.25.2.1 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.

#### 7.25.2.2 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.

### 7.25.2.3 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.

### 7.25.2.4 VTSS\_MTIMER\_TIMEOUT

```
#define VTSS_MTIMER_TIMEOUT(
    pTimer ) (cyg_current_time() > *(pTimer))
```

Timer timeout

Definition at line 55 of file vtss\_os\_ecos.h.

### 7.25.2.5 VTSS\_MTIMER\_CANCEL

```
#define VTSS_MTIMER_CANCEL(
    pTimer )
```

No action in this implementation.

Definition at line 56 of file vtss\_os\_ecos.h.

### 7.25.2.6 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.

### 7.25.2.7 VTSS\_DIV64

```
#define VTSS_DIV64(
    dividend,
    divisor ) ((dividend) / (divisor))
```

support for 64 bit division

Definition at line 73 of file vtss\_os\_ecos.h.

### 7.25.2.8 VTSS\_MOD64

```
#define VTSS_MOD64(
    dividend,
    divisor ) ((dividend) % (divisor))
```

support for 64 bit division

Definition at line 74 of file vtss\_os\_ecos.h.

### 7.25.2.9 VTSS\_LABS

```
#define VTSS_LABS(
    arg ) labs(arg)
```

long to abs

Definition at line 75 of file vtss\_os\_ecos.h.

### 7.25.2.10 VTSS\_LLabs

```
#define VTSS_LLabs(
    arg ) llabs(arg)
```

long long to abs

Definition at line 76 of file vtss\_os\_ecos.h.

### 7.25.2.11 VTSS\_OS\_CTZ

```
#define VTSS_OS_CTZ(
    val32 ) ((val32) == 0 ? 32 : __builtin_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`).

`__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.

### 7.25.2.12 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.

### 7.25.2.13 VTSS\_OS\_MALLOC

```
#define VTSS_OS_MALLOC(
    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.

### 7.25.2.14 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.

### 7.25.2.15 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.

### 7.25.2.16 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.

### 7.25.2.17 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.

### 7.25.2.18 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.

### 7.25.2.19 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.

### 7.25.2.20 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.

### 7.25.2.21 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.

### 7.25.2.22 VTSS\_OS\_BIG\_ENDIAN

```
#define VTSS_OS_BIG_ENDIAN
```

VTSS\_OS\_BIG\_ENDIAN: If undefined, we're running little endian. If defined we're running big endian.

We're running big endian

Definition at line 221 of file vtss\_os\_ecos.h.

### 7.25.2.23 VTSS\_OS\_NTOHL

```
#define VTSS_OS_NTOHL(  
    x ) (x)
```

Convert from network to host order

Definition at line 222 of file vtss\_os\_ecos.h.

### 7.25.2.24 VTSS\_OS\_SCHEDULER\_FLAGS

```
#define VTSS_OS_SCHEDULER_FLAGS cyg_uint32 __attribute__((unused))
```

VTSS\_OS\_SCHEDULER\_FLAGS VTSS\_OS\_SCHEDULER\_LOCK(flags) VTSS\_OS\_SCHEDULER\_UNLOCK(flags)

These functions are called by API code that consists of a user-level part and a deferred interrupt handler part. Only the user-level part will call the VTSS\_OS\_SCHEDULER\_LOCK()//UNLOCK() functions, since it is assumed that the deferred interrupt handler will have atomic access throughout its execution. Each module within the API that contains such functionality will have an option to call either the VTSS\_OS\_SCHEDULER\_(UN)LOCK() functions or the VTSS\_OS\_INTERRUPT\_DISABLE()//RESTORE() functions. The **attribute**((unused)) ensures that we don't get compiler warnings.

Definition at line 248 of file vtss\_os\_ecos.h.

### 7.25.2.25 VTSS\_OS\_SCHEDULER\_LOCK

```
#define VTSS_OS_SCHEDULER_LOCK(  
    flags ) cyg_scheduler_lock(__FILE__, __LINE__)
```

Lock scheduler.

Definition at line 252 of file vtss\_os\_ecos.h.

### 7.25.2.26 VTSS\_OS\_SCHEDULER\_UNLOCK

```
#define VTSS_OS_SCHEDULER_UNLOCK(  
    flags ) cyg_scheduler_unlock(__FILE__, __LINE__)
```

Unlock scheduler.

Definition at line 256 of file vtss\_os\_ecos.h.

### 7.25.2.27 VTSS\_OS\_INTERRUPT\_FLAGS

```
#define VTSS_OS_INTERRUPT_FLAGS NOT_NEEDED
```

VTSS\_OS\_INTERRUPT\_FLAGS [VTSS\\_OS\\_INTERRUPT\\_DISABLE\(flags\)](#) [VTSS\\_OS\\_INTERRUPT\\_RESTORE\(flags\)](#)  
These functions are called by API code that consists of a user-level part and an interrupt handler part executing directly in interrupt context. Only the user-level part will call the [VTSS\\_OS\\_INTERRUPT\\_DISABLE\(\)](#)/[RESTORE\(\)](#) functions, since it is assumed that the interrupt handler will have atomic access throughout its execution. Each module within the API that contains such functionality will have an option to call either the [VTSS\\_OS\\_SCHEDULER\\_\(UN\)LOCK\(\)](#) functions or the [VTSS\\_OS\\_INTERRUPT\\_DISABLE\(\)](#)/[RESTORE\(\)](#) functions. Not needed in eCos, since all interrupt handlers will be called in deferred context.

Definition at line 272 of file vtss\_os\_ecos.h.

### 7.25.2.28 VTSS\_OS\_INTERRUPT\_DISABLE

```
#define VTSS_OS_INTERRUPT_DISABLE( flags ) NOT_NEEDED
```

Disable interrupts.

Definition at line 276 of file vtss\_os\_ecos.h.

### 7.25.2.29 VTSS\_OS\_INTERRUPT\_RESTORE

```
#define VTSS_OS_INTERRUPT_RESTORE( flags ) NOT_NEEDED
```

Enable interrupts.

Definition at line 280 of file vtss\_os\_ecos.h.

## 7.25.3 Typedef Documentation

### 7.25.3.1 vtss\_mtimer\_t

```
typedef cyg_tick_count_t vtss_mtimer_t
```

Timer

Definition at line 53 of file vtss\_os\_ecos.h.

## 7.25.4 Function Documentation

### 7.25.4.1 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*.

**7.25.4.2 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 <code>vtss_mem_flags_t</code> for details.

**Returns**

Pointer to allocated area.

**7.25.4.3 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 <code>vtss_mem_flags_t</code> for details.

## 7.26 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()

#### 7.26.1 Detailed Description

Linux OS API.

This header file describes OS functions for Linux

## 7.26.2 Macro Definition Documentation

### 7.26.2.1 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.

### 7.26.2.2 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.

### 7.26.2.3 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.

#### 7.26.2.4 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.

#### 7.26.2.5 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.

#### 7.26.2.6 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.

### 7.26.2.7 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.

### 7.26.2.8 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.

### 7.26.2.9 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.

### 7.26.2.10 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.

### 7.26.2.11 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.

### 7.26.2.12 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.

### 7.26.2.13 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.

### 7.26.2.14 VTSS\_LABS

```
#define VTSS_LABS( arg ) labs(arg)
```

VTSS\_LABS - perform abs() on long

Definition at line 153 of file vtss\_os\_linux.h.

### 7.26.2.15 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.

### 7.26.2.16 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.

**7.26.2.17 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.

**7.26.2.18 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.

### 7.26.2.19 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.

### 7.26.2.20 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.

## 7.27 vtss\_api/include/vtss\_otn\_api.h File Reference

OTN API.

```
#include <vtss/api/types.h>
```

### 7.27.1 Detailed Description

OTN API.

## 7.28 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\\_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\\_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\_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\_L26\_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 12

## Typedefs

- typedef [u32 vtss\\_packet\\_rx\\_queue\\_size\\_t](#)  
*CPU Rx queue buffer size in bytes.*

## Enumerations

- enum `vtss_packet_filter_t` { `VTSS_PACKET_FILTER_DISCARD`, `VTSS_PACKET_FILTER_TAGGED`, `VTSS_PACKET_FILTER_UNTAGGED` }
 

*CPU filter.*
  - 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_TWOSTEP`,  
`VTSS_PACKET_PTP_ACTION_ONE_AND_TWO_STEP`,  
`VTSS_PACKET_PTP_ACTION_ORIGIN_TIMESTAMP` }
  - enum `vtss_tag_type_t` { `VTSS_TAG_TYPE_UNTAGGED` = 0, `VTSS_TAG_TYPE_C_TAGGED`, `VTSS_TAG_TYPE_S_TAGGED`,  
`VTSS_TAG_TYPE_S_CUSTOM_TAGGED` }
  - enum `vtss_packet_rx_hints_t` { `VTSS_PACKET_RX_HINTS_NONE` = 0x00, `VTSS_PACKET_RX_HINTS_VLAN_TAG_MISMATCH` = 0x01, `VTSS_PACKET_RX_HINTS_VLAN_FRAME_MISMATCH` = 0x02, `VTSS_PACKET_RX_HINTS_VID_MISMATCH` = 0x04 }
 

*Provides additional info on decoded extraction header.*
  - enum `vtss_packet_tx_vstax_t` { `VTSS_PACKET_TX_VSTAX_NONE` = 0, `VTSS_PACKET_TX_VSTAX_BIN`,  
`VTSS_PACKET_TX_VSTAX_SYM` }
- Transmit frames with VStaX header, and if so, how is it specified?*

## Functions

- `vtss_rc vtss_npi_conf_get` (const `vtss_inst_t` inst, `vtss_npi_conf_t` \*const conf)
 

*Get NPI configuration.*
- `vtss_rc vtss_npi_conf_set` (const `vtss_inst_t` inst, const `vtss_npi_conf_t` \*const conf)
 

*Set NPI configuration.*
- `vtss_rc vtss_packet_rx_conf_get` (const `vtss_inst_t` inst, `vtss_packet_rx_conf_t` \*const conf)
 

*Get Packet Rx configuration.*
- `vtss_rc vtss_packet_rx_conf_set` (const `vtss_inst_t` inst, const `vtss_packet_rx_conf_t` \*const conf)
 

*Set CPU Rx queue configuration.*
- `vtss_rc vtss_packet_rx_port_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_packet_rx_port_conf_t` \*const conf)
 

*Get packet configuration for port.*
- `vtss_rc vtss_packet_rx_port_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_packet_rx_port_conf_t` \*const conf)
 

*Set packet configuration for port.*
- `vtss_rc vtss_packet_rx_frame_get` (const `vtss_inst_t` inst, const `vtss_packet_rx_queue_t` queue\_no, `vtss_packet_rx_header_t` \*const header, `u8` \*const frame, const `u32` length)
 

*Copy a received frame from a CPU queue.*
- `vtss_rc vtss_packet_rx_frame_get_raw` (const `vtss_inst_t` inst, `u8` \*const data, const `u32` buflen, `u32` \*const ifhlen, `u32` \*const frrlen)
 

*Copy a received frame from a CPU queue - with IFH.*
- `vtss_rc vtss_packet_rx_frame_discard` (const `vtss_inst_t` inst, const `vtss_packet_rx_queue_t` queue\_no)
 

*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_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.

### 7.28.1 Detailed Description

Packet API.

This header file describes CPU Rx/Tx packet functions.

### 7.28.2 Macro Definition Documentation

### 7.28.2.1 VTSS\_PRIO\_SUPER

```
#define VTSS_PRIO_SUPER VTSS_PRIO_END
```

Super priority

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

### 7.28.2.2 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.

### 7.28.2.3 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.

### 7.28.2.4 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.

### 7.28.2.5 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.

### 7.28.2.6 VTSS\_PACKET\_HDR\_SIZE\_BYTES

```
#define VTSS_PACKET_HDR_SIZE_BYTES VTSS_L26_PACKET_HDR_SIZE_BYTES
```

Maximum header size. This define is only useful if you only compile for one target.

Definition at line 707 of file vtss\_packet\_api.h.

### 7.28.2.7 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.

### 7.28.2.8 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.

### 7.28.2.9 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.

### 7.28.2.10 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.

### 7.28.2.11 VTSS\_PACKET\_TX\_IFH\_MAX

```
#define VTSS_PACKET_TX_IFH_MAX 12
```

Tx IFH byte length (Varies: 8/12 depending on PTP)

Definition at line 2060 of file vtss\_packet\_api.h.

## 7.28.3 Enumeration Type Documentation

### 7.28.3.1 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.

### 7.28.3.2 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.

#### 7.28.3.3 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.

#### 7.28.3.4 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.

#### 7.28.3.5 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.

#### 7.28.3.6 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.

## 7.28.4 Function Documentation

## 7.28.4.1 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.

## 7.28.4.2 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.

**7.28.4.3 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.

**7.28.4.4 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.

**7.28.4.5 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.

**7.28.4.6 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.

**7.28.4.7 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.

**7.28.4.8 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.

**7.28.4.9 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.

**7.28.4.10 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.

**7.28.4.11 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.

**7.28.4.12 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.

**7.28.4.13 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.
-------------	--------------------------

**7.28.4.14 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.

**7.28.4.15 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.

**7.28.4.16 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.

#### 7.28.4.17 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.

#### 7.28.4.18 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-NUL 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-NUL 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-NUL, 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.

**7.28.4.19 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.

**7.28.4.20 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.

**7.28.4.21 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.

**7.28.4.22 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.

**7.28.4.23 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.

**7.28.4.24 vtss\_packet\_dma\_offset()**

```
vtss_rc vtss_packet_dma_offset (
    const vtss_inst_t inst,
    BOOL extraction,
    u32 * offset )
```

Retrive the register offset for extration/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.

## 7.29 vtss\_api/include/vtss\_pcs\_10gbase\_r\_api.h File Reference

PCS\_10BASE\_R API.

```
#include <vtss/api/types.h>
```

### 7.29.1 Detailed Description

PCS\_10BASE\_R API.

## 7.30 vtss\_api/include/vtss\_phy\_10g\_api.h File Reference

10G PHY API

```
#include <vtss/api/types.h>
#include <vtss_misc_api.h>
```

### 7.30.1 Detailed Description

10G PHY API

This header file describes 10G PHY control functions

## 7.31 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\_ACTIPHY\_BIT 0
- #define VTSS\_PHY\_POWER\_DYNAMIC\_BIT 1
- #define VTSS\_PHY\_ACTIPHY\_PWR 100
- #define VTSS\_PHY\_LINK\_DOWN\_PWR 200
- #define VTSS\_PHY\_LINK\_UP\_FULL\_PWR 400
- #define VTSS\_PHY\_RECOV\_CLK1 0
  - PHY active clock out.*
- #define VTSS\_PHY\_RECOV\_CLK2 1
- #define VTSS\_PHY\_RECOV\_CLK\_NUM 2
- #define VTSS\_PHY\_PAGE\_STANDARD 0x0000
- #define VTSS\_PHY\_PAGE\_EXTENDED 0x0001
- #define VTSS\_PHY\_PAGE\_EXTENDED\_2 0x0002
- #define VTSS\_PHY\_PAGE\_EXTENDED\_3 0x0003
- #define VTSS\_PHY\_PAGE\_EXTENDED\_4 0x0004
- #define VTSS\_PHY\_PAGE\_GPIO 0x0010
- #define VTSS\_PHY\_PAGE\_1588 0x1588
- #define VTSS\_PHY\_PAGE\_MACSEC 0x0004
- #define VTSS\_PHY\_PAGE\_TEST 0x2A30
- #define VTSS\_PHY\_PAGE\_TR 0x52B5
- #define VTSS\_PHY\_PAGE\_0x2DAF 0x2DAF
- #define VTSS\_PHY\_REG\_STANDARD (VTSS\_PHY\_PAGE\_STANDARD<<5)
- #define VTSS\_PHY\_REG\_EXTENDED (VTSS\_PHY\_PAGE\_EXTENDED<<5)
- #define VTSS\_PHY\_REG\_GPIO (VTSS\_PHY\_PAGE\_GPIO<<5)
- #define VTSS\_PHY\_REG\_TEST (VTSS\_PHY\_PAGE\_TEST<<5)
- #define VTSS\_PHY\_REG\_TR (VTSS\_PHY\_PAGE\_TR<<5)
- #define VTSS\_PHY\_LINK\_LOS\_EV (1 << 0)
- #define VTSS\_PHY\_LINK\_FFAIL\_EV (1 << 1)
- #define VTSS\_PHY\_LINK\_AMS\_EV (1 << 2)
- #define VTSS\_PHY\_LINK\_SPEED\_STATE\_CHANGE\_EV (1 << 3)
- #define VTSS\_PHY\_LINK\_FDX\_STATE\_CHANGE\_EV (1 << 4)
- #define VTSS\_PHY\_LINK\_AUTO\_NEG\_ERROR\_EV (1 << 5)
- #define VTSS\_PHY\_LINK\_AUTO\_NEG\_COMPLETE\_EV (1 << 6)
- #define VTSS\_PHY\_LINK\_INLINE\_POW\_DEV\_DETECT\_EV (1 << 7)
- #define VTSS\_PHY\_LINK\_SYMBOL\_ERR\_INT\_EV (1 << 8)
- #define VTSS\_PHY\_LINK\_TX\_FIFO\_OVERFLOW\_INT\_EV (1 << 9)
- #define VTSS\_PHY\_LINK\_RX\_FIFO\_OVERFLOW\_INT\_EV (1 << 10)
- #define VTSS\_PHY\_LINK\_FALSE\_CARRIER\_INT\_EV (1 << 11)
- #define VTSS\_PHY\_LINK\_LINK\_SPEED\_DS\_DETECT\_EV (1 << 12)
- #define VTSS\_PHY\_LINK\_MASTER\_SLAVE\_RES\_ERR\_EV (1 << 13)
- #define VTSS\_PHY\_LINK\_RX\_ER\_INT\_EV (1 << 14)
- #define VTSS\_PHY\_LINK\_EXTENDED\_REG\_INT\_EV (1 << 15)
- #define VTSS\_PHY\_LINK\_WAKE\_ON\_LAN\_INT\_EV (1 << 16)
- #define VTSS\_PHY\_LINK\_EXT\_EEE\_WAKE\_ERR\_EV (1 << 17)
- #define VTSS\_PHY\_LINK\_EXT\_EEE\_WAIT\_TS\_EV (1 << 18)
- #define VTSS\_PHY\_LINK\_EXT\_EEE\_WAIT\_RX\_TQ\_EV (1 << 19)
- #define VTSS\_PHY\_LINK\_EXT\_EEE\_LINKFAIL\_EV (1 << 20)
- #define VTSS\_PHY\_LINK\_EXT\_RR\_SW\_COMPL\_EV (1 << 21)
- #define VTSS\_PHY\_LINK\_EXT\_MACSEC\_HOST\_MAC\_EV (1 << 22)
- #define VTSS\_PHY\_LINK\_EXT\_MACSEC\_LINE\_MAC\_EV (1 << 23)
- #define VTSS\_PHY\_LINK\_EXT\_MACSEC\_FC\_BUFF\_EV (1 << 24)
- #define VTSS\_PHY\_LINK\_EXT\_MACSEC\_INGRESS\_EV (1 << 25)

- #define VTSS\_PHY\_LINK\_EXT\_MACSEC\_EGRESS\_EV (1 << 26)
- #define VTSS\_PHY\_LINK\_EXT\_MEM\_INT\_RING\_EV (1 << 27)
- #define MAX\_WOL\_MAC\_ADDR\_SIZE 6
- #define MAX\_WOL\_PASSWD\_SIZE 6
- #define MIN\_WOL\_PASSWD\_SIZE 4

## TypeDefs

- typedef u16 vtss\_phy\_recov\_clk\_t
- typedef u8 vtss\_phy\_led\_intensity  
*PHY led intensity.*
- typedef u32 vtss\_phy\_event\_t  
*PHY interrupt event type.*

## Enumerations

- enum vtss\_phy\_part\_number\_t {
 VTSS\_PHY\_TYPE\_NONE = 0, VTSS\_PHY\_TYPE\_8201 = 8201, VTSS\_PHY\_TYPE\_8204 = 8204, VTSS\_PHY\_TYPE\_8211 = 8211,
 VTSS\_PHY\_TYPE\_8221 = 8221, VTSS\_PHY\_TYPE\_8224 = 8224, VTSS\_PHY\_TYPE\_8234 = 8234, VTSS\_PHY\_TYPE\_8244 = 8244,
 VTSS\_PHY\_TYPE\_8538 = 8538, VTSS\_PHY\_TYPE\_8558 = 8558, VTSS\_PHY\_TYPE\_8574 = 8574, VTSS\_PHY\_TYPE\_8504 = 8504,
 VTSS\_PHY\_TYPE\_8572 = 8572, VTSS\_PHY\_TYPE\_8552 = 8552, VTSS\_PHY\_TYPE\_8501 = 8501, VTSS\_PHY\_TYPE\_8502 = 8502,
 VTSS\_PHY\_TYPE\_7435 = 7435, VTSS\_PHY\_TYPE\_8658 = 8658, VTSS\_PHY\_TYPE\_8601 = 8601, VTSS\_PHY\_TYPE\_8641 = 8641,
 VTSS\_PHY\_TYPE\_7385 = 7385, VTSS\_PHY\_TYPE\_7388 = 7388, VTSS\_PHY\_TYPE\_7389 = 7389, VTSS\_PHY\_TYPE\_7390 = 7390,
 VTSS\_PHY\_TYPE\_7395 = 7395, VTSS\_PHY\_TYPE\_7398 = 7398, VTSS\_PHY\_TYPE\_7500 = 7500, VTSS\_PHY\_TYPE\_7501 = 7501,
 VTSS\_PHY\_TYPE\_7502 = 7502, VTSS\_PHY\_TYPE\_7503 = 7503, VTSS\_PHY\_TYPE\_7504 = 7504, VTSS\_PHY\_TYPE\_7505 = 7505,
 VTSS\_PHY\_TYPE\_7506 = 7506, VTSS\_PHY\_TYPE\_7507 = 7507, VTSS\_PHY\_TYPE\_8634 = 8634, VTSS\_PHY\_TYPE\_8664 = 8664,
 VTSS\_PHY\_TYPE\_8512 = 8512, VTSS\_PHY\_TYPE\_8522 = 8522, VTSS\_PHY\_TYPE\_7420 = 7420, VTSS\_PHY\_TYPE\_8582 = 8582,
 VTSS\_PHY\_TYPE\_8584 = 8584, VTSS\_PHY\_TYPE\_8575 = 8575, VTSS\_PHY\_TYPE\_8564 = 8564, VTSS\_PHY\_TYPE\_8562 = 8562,
 VTSS\_PHY\_TYPE\_8586 = 8586, VTSS\_PHY\_TYPE\_8514 = 8514 }
   
*PHY part ids supported.*
- enum vtss\_phy\_led\_mode\_t {
 LINK\_ACTIVITY, LINK1000\_ACTIVITY, LINK100\_ACTIVITY, LINK10\_ACTIVITY,
 LINK100\_1000\_ACTIVITY, LINK10\_1000\_ACTIVITY, LINK10\_100\_ACTIVITY, LINK100BASE\_FX\_1000BASE\_X\_ACTIVITY,
 DUPLEX\_COLLISION, COLLISION, ACTIVITY, BASE100\_FX\_1000BASE\_X\_FIBER\_ACTIVITY,
 AUTONEGOTIATION\_FAULT, LINK1000BASE\_X\_ACTIVITY, LINK100BASE\_FX\_ACTIVITY, BASE100\_ACTIVITY,
 BASE100\_FX\_ACTIVITY, FORCE\_LED\_OFF, FORCE\_LED\_ON, FAST\_LINK\_FAIL }
   
*PHY LED modes.*
- enum vtss\_phy\_led\_number\_t { LED0, LED1, LED2, LED3 }
   
*List of LED pins per port.*
- enum vtss\_phy\_media\_interface\_t {
 VTSS\_PHY\_MEDIA\_IF\_CU, VTSS\_PHY\_MEDIA\_IF\_SFP\_PASSTHRU, VTSS\_PHY\_MEDIA\_IF\_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 }

- 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` }

- enum `lb_type` { `VTSS_LB_1G_NONE`, `VTSS_LB_FAR_END`, `VTSS_LB_NEAR_END` }
- EEE mode.
- enum `vtss_wol_passwd_len_type_t` { `VTSS_WOL_PASSWD_LEN_4` = 4, `VTSS_WOL_PASSWD_LEN_6` = 6 }
- Internal loop-back type.
- enum `vtss_fefi_mode_t` { `VTSS_100FX_FEFI_NORMAL` = 0, `VTSS_100FX_FEFI_FORCE_SUPPRESS` = 2, `VTSS_100FX_FEFI_FORCE_ENABLE` = 3 }
- Structure for Wake-On-LAN Password Length configuration.
- enum `vtss_fefi_mode_t` { `VTSS_100FX_FEFI_NORMAL` = 0, `VTSS_100FX_FEFI_FORCE_SUPPRESS` = 2, `VTSS_100FX_FEFI_FORCE_ENABLE` = 3 }
- Far-End Failure Indication modes These settings map to PHY Reg23E3, bits 0 and 1.

## Functions

- `vtss_rc vtss_phy_pre_reset` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no)
  - Must be call previous to port PHY Reset (`vtss_phy_reset`).
- `vtss_rc vtss_phy_post_reset` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no)
  - Must be call after port PHY Reset (`vtss_phy_reset`).
- `vtss_rc vtss_phy_pre_system_reset` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no)
  - Must be call before a system reset.
- `vtss_rc vtss_phy_reset` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_reset_conf_t` \*const conf)
  - Reset PHY.
- `vtss_rc vtss_phy_reset_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_reset_conf_t` \*conf)
  - Get reset configuration.
- `vtss_rc vtss_phy_chip_temp_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `i16` \*const temp)
  - Get chip temperature.
- `vtss_rc vtss_phy_chip_temp_init` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no)
  - Init. chip temperature.
- `vtss_rc vtss_phy_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_conf_t` \*const conf)
  - Get PHY configuration.
- `vtss_rc vtss_phy_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_conf_t` \*const conf)
  - Set PHY configuration.
- `vtss_rc vtss_phy_status_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_port_status_t` \*const status)
  - Get PHY status.
- `vtss_rc vtss_phy_cl37_lp_abil_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_port_status_t` \*const status)
  - Get Clause37 Link pArtner's ability.
- `vtss_rc vtss_phy_id_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_type_t` \*phy\_id)
  - Get the PHY type/id. The the Phy ID - can only be used after setting the operating mode.
- `vtss_rc vtss_phy_conf_1g_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_conf_1g_t` \*const conf)
  - Get PHY 1G configuration.
- `vtss_rc vtss_phy_conf_1g_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_conf_1g_t` \*const conf)
  - Set PHY 1G configuration.
- `vtss_rc vtss_phy_status_1g_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_status_1g_t` \*const status)
  - Get PHY 1G status.

- `vtss_rc vtss_phy_power_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_power_conf_t` \*const conf)
 

*Get PHY power configuration.*
- `vtss_rc vtss_phy_power_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_power_conf_t` \*const conf)
 

*Set PHY power configuration.*
- `vtss_rc vtss_phy_power_status_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_power_status_t` \*const status)
 

*Get PHY power status.*
- `vtss_rc vtss_phy_clock_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_recov_clk_t` clock\_port, const `vtss_phy_clock_conf_t` \*const conf)
 

*Set PHY clock configuration.*
- `vtss_rc vtss_phy_clock_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_phy_recov_clk_t` clock\_port, `vtss_phy_clock_conf_t` \*const conf, `vtss_port_no_t` \*const clock\_source)
 

*Get PHY clock configuration.*
- `vtss_rc vtss_phy_i2c_read` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `u8` i2c\_mux, const `u8` i2c\_reg\_addr, const `u8` i2c\_device\_addr, `u8` \*const value, `u8` cnt, `BOOL` word\_access)
 

*I2C read.*
- `vtss_rc vtss_phy_i2c_write` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `u8` i2c\_mux, const `u8` i2c\_reg\_addr, const `u8` i2c\_device\_addr, `u8` \*value, `u8` cnt, `BOOL` word\_access)
 

*I2C writes.*
- `vtss_rc vtss_phy_read` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `u32` addr, `u16` \*const value)
 

*Read value from PHY register.*
- `vtss_rc vtss_phy_read_page` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `u32` page, const `u32` addr, `u16` \*const value)
 

*Read value from PHY register at a specific page. Page register is set to standard page when read is done.*
- `vtss_rc vtss_phy_mmd_read` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `u16` devad, const `u32` addr, `u16` \*const value)
 

*Read value from PHY mmd register.*
- `vtss_rc vtss_phy_mmd_write` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `u16` devad, const `u32` addr, `u16` value)
 

*Write value to PHY mmd register.*
- `vtss_rc vtss_phy_write` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `u32` addr, const `u16` value)
 

*Write value to PHY register.*
- `vtss_rc vtss_phy_write_masked` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `u32` addr, const `u16` value, const `u16` mask)
 

*Write masked value to PHY register.*
- `vtss_rc vtss_phy_write_masked_page` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `u16` page, const `u16` addr, const `u16` value, const `u16` mask)
 

*Write masked value to PHY register and setups the page register.*
- `vtss_rc vtss_phy_gpio_mode` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `u8` gpio\_no, const `vtss_phy_gpio_mode_t` mode)
 

*Configure GPIO mode.*
- `vtss_rc vtss_phy_gpio_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `u8` gpio\_no, `BOOL` \*value)
 

*Get the value from a GPIO pin.*
- `vtss_rc vtss_phy_gpio_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `u8` gpio\_no, `BOOL` value)
 

*Set the value of a GPIO pin.*
- `vtss_rc vtss_phy_veriphy_start` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `u8` mode)
 

*Start VeriPHY.*

- `vtss_rc vtss_phy_veriphy_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_veriphy_result_t *const result)`  
*Get VeriPHY result.*
- `vtss_rc vtss_phy_led_mode_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_led_mode_select_t led_mode_select)`  
*Setting the LEDs blink mode.*
- `vtss_rc vtss_phy_led_intensity_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_led_intensity intensity)`  
*Setting the LEDs intensity.*
- `vtss_rc vtss_phy_led_intensity_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_led_intensity *intensity)`  
*Getting the LEDs intensity.*
- `vtss_rc vtss_phy_enhanced_led_control_init (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_enhanced_led_control_t conf)`  
*Setting the enhanced LED control initial state (Should only be set once at startup)..*
- `vtss_rc vtss_phy_enhanced_led_control_init_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_enhanced_led_control_t *conf)`  
*Getting the enhanced LED control initial state (Should only be set once at startup)..*
- `vtss_rc vtss_phy_coma_mode_disable (const vtss_inst_t inst, const vtss_port_no_t port_no)`  
*Pulling the coma mode pin low.*
- `vtss_rc vtss_phy_coma_mode_enable (const vtss_inst_t inst, const vtss_port_no_t port_no)`  
*Pulling the coma mode pin high (Set in coma mode if no external hardware is controlling the pin)*
- `void vga_adc_debug (const vtss_inst_t inst, u8 vga_adc_pwr, vtss_port_no_t port_no)`  
*debug function for Atom family Rev. A. chips*
- `vtss_rc vtss_phy_port_eee_capable (const vtss_inst_t inst, const vtss_port_no_t port_no, BOOL *eee_capable)`  
*Get information about if a port is EEE capable.*
- `vtss_rc vtss_phy_eee_ena (const vtss_inst_t inst, const vtss_port_no_t port_no, const BOOL enable)`  
*Enabling / Disabling EEE (Energy Efficient Ethernet)*
- `vtss_rc vtss_phy_eee_conf_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_eee_conf_t *conf)`  
*Getting the current EEE (Energy Efficient Ethernet) configuration.*
- `vtss_rc vtss_phy_eee_conf_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_eee_conf_t conf)`  
*Setting the EEE (Energy Efficient Ethernet) configuration.*
- `vtss_rc vtss_phy_eee_power_save_state_get (const vtss_inst_t inst, const vtss_port_no_t port_no, BOOL *rx_in_power_save_state, BOOL *tx_in_power_save_state)`  
*Getting the if phy is currently powered save mode due to EEE.*
- `vtss_rc vtss_phy_eee_link_partner_advertisements_get (const vtss_inst_t inst, const vtss_port_no_t port_no, u8 *advertisement)`  
*Getting the EEE advertisement.*
- `vtss_rc vtss_phy_event_enable_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_event_t ev_mask, const BOOL enable)`  
*Enabling / Disabling of events.*
- `vtss_rc vtss_phy_event_enable_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_event_t *ev_mask)`  
*Getting current interrupt event state.*
- `vtss_rc vtss_phy_event_poll (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_event_t *const ev_mask)`  
*Polling for active events.*
- `vtss_rc vtss_squelch_workaround (const vtss_inst_t inst, const vtss_port_no_t port_no, const BOOL enable)`  
*Function for enabling/disabling squelch work around.*

- `vtss_rc vtss_phy_csr_wr` (const `vtss_inst_t` inst, const `u16` page, const `vtss_port_no_t` port\_no, const `u16` target, const `u32` csr\_reg\_addr, const `u32` value)
 

*Function for writing to CSR registers.*
- `vtss_rc vtss_phy_csr_rd` (const `vtss_inst_t` inst, const `u16` page, const `vtss_port_no_t` port\_no, const `u16` target, const `u32` csr\_reg\_addr, `u32` \*value)
 

*Function for writing to CSR registers.*
- `vtss_rc vtss_phy_statistic_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_statistic_t` \*statistics)
 

*debug function for getting phy statistics.*
- `vtss_rc vtss_phy_do_page_chk_set` (const `vtss_inst_t` inst, const `BOOL` enable)
 

*Debug function for enabling check of page register for all phy register accesses.*
- `vtss_rc vtss_phy_do_page_chk_get` (const `vtss_inst_t` inst, `BOOL` \*enable)
 

*Debug function for getting if check of page register is enabled.*
- `vtss_rc vtss_phy_loopback_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_loopback_t` loopback)
 

*Debug function for setting phy internal loopback.*
- `vtss_rc vtss_phy_loopback_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_phy_loopback_t` \*loopback)
 

*Debug function for getting the current phy internal loopback.*
- `vtss_rc vtss_phy_is_8051_crc_ok` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `u16` code\_length, `u16` expected\_crc)
 

*Debug function for checking if the phy firmware is loaded correctly.*
- `vtss_rc vtss_phy_cfg_ob_post0` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `u16` value)
 

*Debug function for setting the ob post0 patch.*
- `vtss_rc vtss_phy_cfg_ib_cterm` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `u8` ib\_cterm\_ena, const `u8` ib\_eq\_mode)
 

*Debug function for setting the ib cterm patch.*
- `vtss_rc vtss_phy_atom12_patch_settings_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `u8` \*mcb\_bus, `u8` \*cfg\_buf, `u8` \*stat\_buf)
 

*Debug function for getting PHY setting set by the micro patches.*
- `void vtss_phy_reg_decode_status` (`vtss_port_no_t` port\_no, `u16` lp\_auto\_neg\_advertisement\_reg, `u16` lp\_1000base\_t\_status\_reg, `u16` mii\_status\_reg, const `vtss_phy_conf_t` phy\_setup, `vtss_port_status_t` \*const status)
 

*Function for updating the status via the result from PHY registers.*
- `vtss_rc vtss_phy_flowcontrol_decode_status` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `u16` lp\_auto\_neg\_advertisement\_reg, const `vtss_phy_conf_t` phy\_setup, `vtss_port_status_t` \*const status)
 

*Function for finding flow control status based upon configuration and PHY registers.*
- `vtss_rc vtss_phy_debug_stat_print` (const `vtss_inst_t` inst, const `vtss_debug_printf_t` pr, const `vtss_port_no_t` port\_no, const `BOOL` print\_hdr)
 

*debug function for printing PHY statistics*
- `vtss_rc vtss_phy_warm_start_failed_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no)
 

*Function for checking if any issue were seen during warm-start.*
- `vtss_rc vtss_phy_debug_physinfo_print` (const `vtss_inst_t` inst, const `vtss_debug_printf_t` pr, const `vtss_port_no_t` port\_no, const `BOOL` print\_hdr)
 

*debug function for printing some of the internal PHY state/configurations*
- `vtss_rc vtss_phy_debug_register_dump` (const `vtss_inst_t` inst, const `vtss_debug_printf_t` pr, `BOOL` clear, const `vtss_port_no_t` port\_no)
 

*debug function for printing some of the internal PHY state/configurations*
- `vtss_rc vtss_phy_detect_base_ports` (const `vtss_inst_t` inst)
 

*Function for making getting the API updated with base port number for all ports. MUST not be called unless all ports are detected.*
- `vtss_rc vtss_phy_ext_connector_loopback` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `BOOL` lpback)
 

*lpback)*

- *Function for configuring External Connector Loopback.*
- **vtss\_rc vtss\_phy\_serdes\_sgmii\_loopback** (const **vtss\_inst\_t** inst, const **vtss\_port\_no\_t** port\_no, const **u16** mode)
  - *Function for configuring MAC-SerDes(SGMII) Loopback.*
  - **vtss\_rc vtss\_phy\_serdes\_fmedia\_loopback** (const **vtss\_inst\_t** inst, const **vtss\_port\_no\_t** port\_no, const **u16** mode)
    - *Function for configuring Fibre-Media SerDes Loopback.*
    - **vtss\_rc vtss\_phy\_debug\_reddump\_print** (const **vtss\_inst\_t** inst, const **vtss\_debug\_printf\_t** pr, const **vtss\_port\_no\_t** port\_no, const **vtss\_port\_no\_t** page\_no, const **BOOL** print\_hdr)
      - *debug function for printing some of the internal PHY Registers*
    - **vtss\_rc vtss\_phy\_wol\_enable** (const **vtss\_inst\_t** inst, const **vtss\_port\_no\_t** port\_no, const **BOOL** enable)
      - *function to Enable or Disable WOL by enabling or disabling the interrupt*
    - **vtss\_rc vtss\_phy\_wol\_conf\_get** (const **vtss\_inst\_t** inst, const **vtss\_port\_no\_t** port\_no, **vtss\_phy\_wol\_conf\_t** \*const conf)
      - *function to Get Wake-On-LAN configuration*
    - **vtss\_rc vtss\_phy\_wol\_conf\_set** (const **vtss\_inst\_t** inst, const **vtss\_port\_no\_t** port\_no, const **vtss\_phy\_wol\_conf\_t** \*const conf)
      - *function to Set Wake-On-LAN configuration*
  - **vtss\_rc vtss\_phy\_patch\_settings\_get** (const **vtss\_inst\_t** inst, const **vtss\_port\_no\_t** port\_no, **u8** \*mcb\_bus, **u8** \*cfg\_buf, **u8** \*stat\_buf)
    - *Debug function for getting PHY setting set by the micro patches.*
- **vtss\_rc vtss\_phy\_serdes6g\_rcpll\_status\_get** (const **vtss\_inst\_t** inst, const **vtss\_port\_no\_t** port\_no, **vtss\_rcpll\_status\_t** \*rcpll\_status)
  - *Debug function for getting PHY Serdes6G RC-PLL status.*
- **vtss\_rc vtss\_phy\_serdes1g\_rcpll\_status\_get** (const **vtss\_inst\_t** inst, const **vtss\_port\_no\_t** port\_no, **vtss\_rcpll\_status\_t** \*rcpll\_status)
  - *Debug function for getting PHY Serdes1G RC-PLL status.*
- **vtss\_rc vtss\_phy\_lcpll\_status\_get** (const **vtss\_inst\_t** inst, const **vtss\_port\_no\_t** port\_no, **vtss\_lcpll\_status\_t** \*lcpll\_status)
  - *Debug function for getting PHY LC-PLL status.*
- **vtss\_rc vtss\_phy\_reset\_lcpll** (const **vtss\_inst\_t** inst, const **vtss\_port\_no\_t** port\_no)
  - *Debug function for Resetting the LCPLL for the PHY.*
- **vtss\_rc vtss\_phy\_sd6g\_ob\_post\_rd** (**vtss\_inst\_t** inst, const **vtss\_port\_no\_t** port\_no, **u8** \*ob\_post0, **u8** \*ob\_post1)
  - *Debug function for reading the 6G SerDes ob\_post0 and ob\_post1 values.*
- **vtss\_rc vtss\_phy\_sd6g\_ob\_post\_wr** (**vtss\_inst\_t** inst, const **vtss\_port\_no\_t** port\_no, const **u8** ob\_post0, const **u8** ob\_post1)
  - *Debug function for modifying the 6G SerDes ob\_post0 and ob\_post1 values.*
- **vtss\_rc vtss\_phy\_sd6g\_ob\_lev\_rd** (**vtss\_inst\_t** inst, const **vtss\_port\_no\_t** port\_no, **u8** \*ob\_level)
  - *Debug function for reading the 6G SerDes ob\_level value.*
- **vtss\_rc vtss\_phy\_sd6g\_ob\_lev\_wr** (**vtss\_inst\_t** inst, const **vtss\_port\_no\_t** port\_no, const **u8** ob\_level)
  - *Debug function for modifying the 6G SerDes ob\_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)*
  - `vtss_rc vtss_phy_sd6g_mac_serdes_conf (const vtss_inst_t inst, const vtss_port_no_t port_no)`

*Function for Configuring MAC i/f 6G serdes (applicable only on VIPER,ELISE)*

### 7.31.1 Detailed Description

PHY API.

This header file describes PHY control functions

### 7.31.2 Macro Definition Documentation

#### 7.31.2.1 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.

#### 7.31.2.2 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.

### 7.31.2.3 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.

### 7.31.2.4 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.

### 7.31.2.5 VTSS\_PHY\_ACTIPHY\_PWR

```
#define VTSS_PHY_ACTIPHY_PWR 100
```

ActiPHY power status

Definition at line 592 of file vtss\_phy\_api.h.

### 7.31.2.6 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.

### 7.31.2.7 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.

### 7.31.2.8 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.

### 7.31.2.9 VTSS\_PHY\_RECov\_CLK2

```
#define VTSS_PHY_RECov_CLK2 1
```

RCVRD\_CLK2

Definition at line 618 of file vtss\_phy\_api.h.

### 7.31.2.10 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.

### 7.31.2.11 VTSS\_PHY\_PAGE\_STANDARD

```
#define VTSS_PHY_PAGE_STANDARD 0x0000
```

Standard registers

Definition at line 740 of file vtss\_phy\_api.h.

### 7.31.2.12 VTSS\_PHY\_PAGE\_EXTENDED

```
#define VTSS_PHY_PAGE_EXTENDED 0x0001
```

Extended registers

Definition at line 741 of file vtss\_phy\_api.h.

### 7.31.2.13 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.

### 7.31.2.14 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.

### 7.31.2.15 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.

### 7.31.2.16 VTSS\_PHY\_PAGE\_GPIO

```
#define VTSS_PHY_PAGE_GPIO 0x0010
```

GPIO registers

Definition at line 745 of file vtss\_phy\_api.h.

### 7.31.2.17 VTSS\_PHY\_PAGE\_1588

```
#define VTSS_PHY_PAGE_1588 0x1588
```

1588 (PTP) registers

Definition at line 746 of file vtss\_phy\_api.h.

### 7.31.2.18 VTSS\_PHY\_PAGE\_MACSEC

```
#define VTSS_PHY_PAGE_MACSEC 0x0004
```

MACSEC page

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

### 7.31.2.19 VTSS\_PHY\_PAGE\_TEST

```
#define VTSS_PHY_PAGE_TEST 0x2A30
```

Test registers

Definition at line 748 of file vtss\_phy\_api.h.

### 7.31.2.20 VTSS\_PHY\_PAGE\_TR

```
#define VTSS_PHY_PAGE_TR 0x52B5
```

Token ring registers

Definition at line 749 of file vtss\_phy\_api.h.

### 7.31.2.21 VTSS\_PHY\_PAGE\_0x2DAF

```
#define VTSS_PHY_PAGE_0x2DAF 0x2DAF
```

0x2DAF registers

Definition at line 750 of file vtss\_phy\_api.h.

### 7.31.2.22 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.

### 7.31.2.23 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.

### 7.31.2.24 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.

### 7.31.2.25 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.

### 7.31.2.26 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.

### 7.31.2.27 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.

**7.31.2.28 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.

**7.31.2.29 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.

**7.31.2.30 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.

**7.31.2.31 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.

**7.31.2.32 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.

**7.31.2.33 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.

**7.31.2.34 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.

**7.31.2.35 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.

**7.31.2.36 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.

**7.31.2.37 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.

**7.31.2.38 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.

**7.31.2.39 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.

**7.31.2.40 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.

**7.31.2.41 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.

**7.31.2.42 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.

**7.31.2.43 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.

**7.31.2.44 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.

**7.31.2.45 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.

**7.31.2.46 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.

**7.31.2.47 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.

**7.31.2.48 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.

**7.31.2.49 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.

**7.31.2.50 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.

**7.31.2.51 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.

**7.31.2.52 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.

**7.31.2.53 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.

**7.31.2.54 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.

**7.31.2.55 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.

**7.31.2.56 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.

**7.31.2.57 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.

---

### 7.31.3 Typedef Documentation

#### 7.31.3.1 `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.

#### 7.31.3.2 `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.

### 7.31.4 Enumeration Type Documentation

#### 7.31.4.1 `vtss_phy_led_mode_t`

```
enum vtss_phy_led_mode_t
```

PHY LED modes.

Enumerator

<code>LINK1000_ACTIVITY</code>	No link in any speed on any media interface./Valid link at any speed on any media interface. Blink or pulse-stretch = Valid link at any speed on any media interface with activity present.
<code>LINK100_ACTIVITY</code>	No link in 1000BASE-T/Valid 1000BASE-T link. Blink or pulse-stretch = Valid 1000BASE-T link with activity present
<code>LINK10_ACTIVITY</code>	No link in 100BASE-T/Valid 100BASE-T link. Blink or pulse-stretch = Valid 100BASE-T link with activity present
<code>LINK100_1000_ACTIVITY</code>	No link in 10BASE-T/Valid 10BASE-T link. Blink or pulse-stretch = Valid 10BASE-T link with activity present
<code>LINK10_1000_ACTIVITY</code>	No link in 100BASE-T, 1000BASE-FX, or 1000BASE-TX/Valid 100BASE-T, 1000BASE-FX, or 1000BASE-TX link.Blink or pulse-stretch = Valid 100BASE-T, 1000BASE-FX, or 1000BASE-TX link with activity present.

Generated by Doxygen

## Enumerator

LINK10_100_ACTIVITY	No link in 10BASE-T, 1000BASE-FX, or 1000BASE-TX/Valid 10BASE-T, 1000BASE-FX, or 1000BASE-TX link.Blink or pulse-stretch = Valid 10BASE-T, 1000BASE-FX, or 1000BASE-TX link with activity present.
LINK100BASE_FX_1000BASE_X_ACTIVITY	No link in 10BASE-T, 100BASE-FX, or 100BASE-TX/Valid 10BASE-T, 100BASE-FX, or 100BASE-TX link.Blink or pulse-stretch = Valid 10BASE-T, 100BASE-FX, or 100BASE-TX link with activity present.
DUPLEX_COLLISION	No link in 100BASE-FX or 1000BASE-X/ Valid 100BASE-FX or 1000BASE-X link. Blink or pulse-stretch = Valid 100BASE-FX or 1000BASE-X link with activity present.
COLLISION	Link established in half-duplex mode, or no link established. Link established in full-duplex mode.Blink or pulse-stretch = Link established in half-duplex mode but collisions are present
ACTIVITY	No collision detected.Blink or pulse-stretch = Collision detected.
BASE100_FX_1000BASE_X_FIBER_ACTIVITY	No activity present.Blink or pulse-stretch = Activity present
AUTONEGOTIATION_FAULT	No 100BASE-FX or 1000BASE-X activity present. Blink or pulse-stretch = 100BASE-FX or 1000BASE-X activity present
LINK1000BASE_X_ACTIVITY	No autonegotiation fault present., Autonegotiation fault occurred.
LINK100BASE_FX_ACTIVITY	No link in 1000BASE-X. Valid 1000BASE-X link.
BASE100_ACTIVITY	No link in 100BASE-FX.
BASE100_FX_ACTIVITY	No 1000BASE-X activity present.Blink or pulse-stretch = 1000BASE-X activity present.
FORCE_LED_OFF	No 100BASE-FX activity present, Blink or pulse-stretch = 100BASE-FX activity present.
FORCE_LED_ON	De-asserts the LED
FAST_LINK_FAIL	Asserts the LED

Definition at line 102 of file vtss\_phy\_api.h.

## 7.31.4.2 vtss\_phy\_media\_interface\_t

```
enum vtss_phy_media_interface_t
```

PHY media interface type.

## Enumerator

VTSS_PHY_MEDIA_IF_CU	Copper Interface
VTSS_PHY_MEDIA_IF_SFP_PASSTHRU	Fiber/Cu SFP Pass-thru
VTSS_PHY_MEDIA_IF_FI_1000BX	1000Base-X
VTSS_PHY_MEDIA_IF_FI_100FX	100Base-FX

**Enumerator**

VTSS_PHY_MEDIA_IF_AMS CU_PASSTHRU	AMS - Cat5/SerDes/CuSFP passthru - Note the phy mode must be set to VTSS_PHY_MODE_ANEG
VTSS_PHY_MEDIA_IF_AMS FI_PASSTHRU	AMS - Fiber passthru - Note the phy mode must be set to VTSS_PHY_MODE_ANEG
VTSS_PHY_MEDIA_IF_AMS CU_1000BX	AMS - Cat5/1000BX/CuSFP
VTSS_PHY_MEDIA_IF_AMS CU_100FX	AMS - Cat5/100FX/CuSFP

Definition at line 161 of file vtss\_phy\_api.h.

**7.31.4.3 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.

**7.31.4.4 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.

#### 7.31.4.5 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.

#### 7.31.4.6 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.

#### 7.31.4.7 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.

#### 7.31.4.8 `vtss_phy_fast_link_fail_t`

enum `vtss_phy_fast_link_fail_t`

PHY fast link failure pin enable/disable.

Enumerator

<code>VTSS_PHY_FAST_LINK_FAIL_ENABLE</code>	Enable fast link failure pin
<code>VTSS_PHY_FAST_LINK_FAIL_DISABLE</code>	Disable fast link failure pin

Definition at line 325 of file vtss\_phy\_api.h.

#### 7.31.4.9 `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.

#### 7.31.4.10 `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.

#### 7.31.4.11 `vtss_phy_mac_serdes_pcs_sgmii_pre`

enum `vtss_phy_mac_serdes_pcs_sgmii_pre`

PHY MEDIA SerDes PCS Remote Fault Indication, See Clause 37, Table 37-3.

## Enumerator

VTSS_PHY_MAC_SERD_PCS_SGMII_IN_PRE_↔ NONE	MAC SerDes PCS Control, SGMII Input Preamble for 100BaseX - No Preamble Req'd
VTSS_PHY_MAC_SERD_PCS_SGMII_IN_PRE_↔ ONE	MAC SerDes PCS Control, SGMII Input Preamble for 100BaseX - One-Byte Preamble Req'd
VTSS_PHY_MAC_SERD_PCS_SGMII_IN_PRE_↔ TWO	MAC SerDes PCS Control, SGMII Input Preamble for 100BaseX - Two-Byte Preamble Req'd
VTSS_PHY_MAC_SERD_PCS_SGMII_IN_PRE_↔ RSVD	MAC SerDes PCS Control, SGMII Input Preamble for 100BaseX - Reserved

Definition at line 343 of file vtss\_phy\_api.h.

## 7.31.4.12 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_L↔ INK_FAIL	Media SerDes PCS Control, Most Recent Clause 37 ANEG Exchg
VTSS_PHY_MEDIA_SERD_PCS_Rem_Fault_A↔ NEG_ERROR	Media SerDes PCS Control, Most Recent Clause 37 ANEG Exchg

Definition at line 367 of file vtss\_phy\_api.h.

## 7.31.4.13 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.

#### 7.31.4.14 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.

#### 7.31.4.15 vtss\_phy\_freq\_t

enum [vtss\\_phy\\_freq\\_t](#)

PHY clock frequencies.

Enumerator

VTSS_PHY_FREQ_25M	25 MHz
VTSS_PHY_FREQ_125M	125 MHz
VTSS_PHY_FREQ_3125M	31.25 MHz This is only valid on ATOM family - NOT Enzo

Definition at line 632 of file vtss\_phy\_api.h.

#### 7.31.4.16 vtss\_phy\_clk\_squelch

enum [vtss\\_phy\\_clk\\_squelch](#)

PHY clock squelch levels.

## Enumerator

VTSS_PHY_CLK_SQUELCH_MAX	Automatically squelch clock to low when the link is not up, is unstable, is up in a mode that does not support the generation of arecovered clock (1000BASE-T master or 10BASE-T), or is up in EEE mode (100BASE-TX or 1000BASE-T slave).
VTSS_PHY_CLK_SQUELCH_MED	Same as VTSS_PHY_CLK_SQUELCH_MAX except that the clock is also generated in 1000BASE-T master and 10BASE-T link-up modes. This mode also generates a recovered clock output in EEE mode during reception of LP_IDLE.
VTSS_PHY_CLK_SQUELCH_MIN	Squelch only when the link is not up
VTSS_PHY_CLK_SQUELCH_NONE	Disable clock squelch.

Definition at line 639 of file vtss\_phy\_api.h.

7.31.4.17 `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.

7.31.4.18 `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.

#### 7.31.4.19 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.

#### 7.31.4.20 vtss\_fefi\_mode\_t

```
enum vtss_fefi_mode_t
```

Far-End Failure Indication modes These settings map to PHY Reg23E3, bits 0 and 1.

Enumerator

VTSS_100FX_FEFI_NORMAL	Normal FEFI Operation, as specified by Reg23E3.1
VTSS_100FX_FEFI_FORCE_SUPPRESS	Force FEFI, as specified by Reg23E3.0
VTSS_100FX_FEFI_FORCE_ENABLE	Force FEFI, as specified by Reg23E3.0

Definition at line 1900 of file vtss\_phy\_api.h.

### 7.31.5 Function Documentation

#### 7.31.5.1 vtss\_phy\_pre\_reset()

```
vtss_rc vtss_phy_pre_reset (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no )
```

Must be call previous to port PHY Reset (vtss\_phy\_reset).

**Parameters**

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number (MUST be the first port for the chip).

**Returns**

Return code.

**7.31.5.2 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.

**7.31.5.3 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.

#### 7.31.5.4 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.

#### 7.31.5.5 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.

#### 7.31.5.6 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.

**7.31.5.7 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.

**7.31.5.8 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.

**7.31.5.9 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.

**7.31.5.10 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.

### 7.31.5.11 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.

### 7.31.5.12 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.

### 7.31.5.13 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.

**7.31.5.14 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.

**7.31.5.15 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.

**7.31.5.16 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.

**7.31.5.17 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.

## 7.31.5.18 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.

## 7.31.5.19 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.

## 7.31.5.20 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.

### 7.31.5.21 vtss\_phy\_i2c\_read()

```
vtss_rc vtss_phy_i2c_read (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const u8 i2c_mux,
    const u8 i2c_reg_addr,
    const u8 i2c_device_addr,
    u8 *const value,
    u8 cnt,
    BOOL word_access )
```

I2C read.

#### Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>i2c_mux</i>	[IN] The i2c clock mux
<i>i2c_reg_addr</i>	[IN] The i2c register address to access.
<i>i2c_device_addr</i>	[IN] The i2c address of the device to access.
<i>value</i>	[OUT] Pointer to where array which in going to contain the values read.
<i>cnt</i>	[IN] The number of registers to read. Note: The reg_addr is incremented by 1 for each of the read counts. If you want to read 16 bits registers (2 times 8 bits from the same register address), you need to do that by calling the vtss_phy_i2c_read twice, and not use the cnt (set cnt to 1).
<i>word_access</i>	[IN] Set to TRUE if the register data width is 16bit. FALSE = 8 bits data width

**Returns**

Return code.

**7.31.5.22 vtss\_phy\_i2c\_write()**

```
vtss_rc vtss_phy_i2c_write (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const u8 i2c_mux,
    const u8 i2c_reg_addr,
    const u8 i2c_device_addr,
    u8 * value,
    u8 cnt,
    BOOL word_access )
```

I2C writes.

**Parameters**

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>i2c_mux</i>	[IN] The i2c clock mux
<i>i2c_reg_addr</i>	[IN] The i2c register address to access.
<i>i2c_device_addr</i>	[IN] The i2c address of the device to access.
<i>value</i>	[IN] Pointer to where array containing the values to write.
<i>cnt</i>	[IN] The number of registers to write. Note: The reg_addr is incremented by 1 for each of the write counts. If you want to write 16 bites registers (2 times 8 bits to the same register address), you need to do that by calling the vtss_phy_i2c_write twice, and not use the cnt (set cnt to 1).
<i>word_access</i>	[IN] Set to TRUE if the register data width is 16bit. FALSE = 8 bits data width

**Returns**

Return code.

**7.31.5.23 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.

**7.31.5.24 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.

**7.31.5.25 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.

**7.31.5.26 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.

**7.31.5.27 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.

**7.31.5.28 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.

**7.31.5.29 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.

**7.31.5.30 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.

**7.31.5.31 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.

**7.31.5.32 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.

**7.31.5.33 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.

**7.31.5.34 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.

**7.31.5.35 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.

### 7.31.5.36 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.

### 7.31.5.37 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.

### 7.31.5.38 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.

**7.31.5.39 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.

**7.31.5.40 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.

**7.31.5.41 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.

**7.31.5.42 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).

**7.31.5.43 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.

#### 7.31.5.44 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.

#### 7.31.5.45 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.

**7.31.5.46 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.

**7.31.5.47 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.

**7.31.5.48 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.

**7.31.5.49 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.

### 7.31.5.50 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.

### 7.31.5.51 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.

### 7.31.5.52 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

**7.31.5.53 vtss\_phy\_csr\_wr()**

```
vtss_rc vtss_phy_csr_wr (
    const vtss_inst_t inst,
    const u16 page,
    const vtss_port_no_t port_no,
    const u16 target,
    const u32 csr_reg_addr,
    const u32 value )
```

Function for writing to CSR registers.

**Parameters**

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] The port in question
<i>page</i>	[IN] The page for the CSR register access e.g. VTSS_PHY_PAGE_1588 or VTSS_PHY_PAGE_MACSEC
<i>target</i>	[IN] The CSR target
<i>csr_reg_addr</i>	[IN] The CSR register to write
<i>value</i>	[IN] The value to write

**Returns**

VTSS\_RC\_OK when new access can be done - VTSS\_RC\_ERROR if something went wrong and access was never granted.

**7.31.5.54 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.

**7.31.5.55 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

**7.31.5.56 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.

**7.31.5.57 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.

**7.31.5.58 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

**7.31.5.59 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

### 7.31.5.60 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

### 7.31.5.61 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

**7.31.5.62 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

**7.31.5.63 vtss\_phy\_atom12\_patch\_settings\_get()**

```
vtss_rc vtss_phy_atom12_patch_settings_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    u8 * mcb_bus,
    u8 * cfg_buf,
    u8 * stat_buf )
```

Debug function for getting PHY setting set by the micro patches.

**Parameters**

<i>inst</i>	[IN] Target instance reference.
-------------	---------------------------------

**Parameters**

<i>port_no</i>	[IN] PHY port within the chip.
<i>mcb_bus</i>	[INOUT] If set to 2 the returned data is for LCPLL/RComp else the mcb_bus is returning 0 if the data is for 1G, and returning 1 if the data is for 6G .
<i>cfg_buf</i>	[IN] Pointer to array where to put the current settings.
<i>stat_buf</i>	[IN] Pointer to array where to put the current status.

**Returns**

Return code. VTSS\_RC\_OK if patch were updated correct else VTSS\_RC\_ERROR

**7.31.5.64 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

**7.31.5.65 vtss\_phy\_flowcontrol\_decode\_status()**

```
vtss_rc vtss_phy_flowcontrol_decode_status (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    u16 lp_auto_neg_advertisement_reg,
    const vtss_phy_conf_t phy_setup,
    vtss_port_status_t *const status )
```

Function for finding flow control status based upon configuration and PHY registers.

**Parameters**

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] The port in question.
<i>lp_auto_neg_advertisement_reg</i>	[IN] The value from the register containing the Link partners auto negotiation advertisement (Standard page 5)
<i>phy_setup</i>	[IN] The phy configuration setup
<i>status</i>	[INOUT] Pointer to where to put the result *

**Returns**

Return code. VTSS\_RC\_OK if not errors were seen during warm-start else VTSS\_RC\_ERROR.

**7.31.5.66 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.

**7.31.5.67 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.

**7.31.5.68 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.

**7.31.5.69 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.

**7.31.5.70 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.

**7.31.5.71 vtss\_phy\_ext\_connector\_loopback()**

```
vtss_rc vtss_phy_ext_connector_loopback (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const BOOL lpback )
```

Function for configuring External Connector Loopback.

**Parameters**

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] The port in question.
<i>lpback</i>	[IN] TRUE=Loopback ON, FALSE=Loopback OFF

**Returns**

Return code. VTSS\_RC\_OK if not errors were seen during warm-start else VTSS\_RC\_ERROR.

**7.31.5.72 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 not errors were seen during warm-start else VTSS\_RC\_ERROR.

### 7.31.5.73 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.

### 7.31.5.74 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.

**7.31.5.75 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.

**7.31.5.76 vtss\_phy\_wol\_conf\_get()**

```
vtss_rc vtss_phy_wol_conf_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_phy_wol_conf_t *const conf )
```

function to Get Wake-On-LAN configuration

**Parameters**

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port in question
<i>conf</i>	[IN] Ptr to WoL Structure <a href="#">vtss_phy_wol_conf_t</a> to be filled out by API

**Returns**

Return code. VTSS\_RC\_OK if no errors.

**7.31.5.77 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.

**7.31.5.78 vtss\_phy\_patch\_settings\_get()**

```
vtss_rc vtss_phy_patch_settings_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    u8 * mcb_bus,
    u8 * cfg_buf,
    u8 * stat_buf )
```

Debug function for getting PHY setting set by the micro patches.

**Parameters**

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] PHY port within the chip.

**Parameters**

<i>mcb_bus</i>	[INOUT] If set to 2 the returned data is for LCPLL/RComp else the 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

**7.31.5.79 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

**7.31.5.80 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

**7.31.5.81 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

**7.31.5.82 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.

**7.31.5.83 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.

**7.31.5.84 vtss\_phy\_sd6g\_ob\_post\_wr()**

```
vtss_rc vtss_phy_sd6g_ob_post_wr (
    vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const u8 ob_post0,
    const u8 ob_post1 )
```

Debug function for modifying the 6G SerDes ob\_post0 and ob\_post1 values.

**Parameters**

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] PHY port within the chip.
<i>ob_post0</i>	[IN] Modify ob_post0 settings in serdes6g_ob_cfg for 6G SerDes Macro (See TN1052), 6 bits, default:0, Customizable.
<i>ob_post1</i>	[IN] Modify ob_post1 settings in serdes6g_ob_cfg for 6G SerDes Macro (See TN1052), 5 bits, default:0, Do not change.

**Returns**

Return code. VTSS\_RC\_OK if all Ok VTSS\_RC\_ERROR if and error occurred.

**7.31.5.85 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.

**7.31.5.86 vtss\_phy\_sd6g\_ob\_lev\_wr()**

```
vtss_rc vtss_phy_sd6g_ob_lev_wr (
    vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const u8 ob_level )
```

Debug function for modifying the 6G SerDes ob\_lev value.

**Parameters**

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] PHY port within the chip.
<i>ob_level</i>	[IN] Modify ob_lev settings in serdes6g_ob_cfg1 for 6G SerDes Macro (See TN1052), 6 bits, default:0x18, Customizeable Amplitude Control.

**Returns**

Return code. VTSS\_RC\_OK if all Ok VTSS\_RC\_ERROR if and error occurred.

### 7.31.5.87 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.

### 7.31.5.88 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.

### 7.31.5.89 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.

**7.31.5.90 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.

**7.31.5.91 vtss\_phy\_mse\_100m\_get()**

```
vtss_rc vtss_phy_mse_100m_get (
    vtss_inst_t inst,
    const vtss_port_no_t port_no,
    u32 * mse )
```

Function to get the Mean Squared Error (MSE), ie. Noise floor for 100m.

**Parameters**

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] PHY port within the chip.
<i>mse</i>	[OUT] PHY port MSE Value In order to convert the returned mse value to dB, do the following This computation would be required for each channel pair mse_dbl = mse / (1024 * 2048); Note: Convert to Double and Scale by $2^{21} = 1024 * 2048$ mse_dbl = $20 * \log_{10}(\text{mse\_dbl})$ ; Note: Convert to dB Nominal Computed Values for mse_dbl are: -22dB to -31dB

**Returns**

Return code. VTSS\_RC\_OK if all Ok VTSS\_RC\_ERROR if and error occurred.

**7.31.5.92 vtss\_phy\_mse\_1000m\_get()**

```
vtss_rc vtss_phy_mse_1000m_get (
    vtss_inst_t inst,
    const vtss_port_no_t port_no,
    u32 * mseA,
    u32 * mseB,
    u32 * mseC,
    u32 * mseD )
```

Function to get the Mean Squared Error (MSE), ie. Noise floor for 1000m.

**Parameters**

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] PHY port within the chip.
<i>mseA</i>	[OUT] PHY port MSE Chan A Value
<i>mseB</i>	[OUT] PHY port MSE Chan B Value
<i>mseC</i>	[OUT] PHY port MSE Chan C Value
<i>mseD</i>	[OUT] PHY port MSE Chan D Value In order to convert the returned mse value to dB, do the following This computation would be required for each channel pair $\text{mse\_dbl} = \text{mse} / (1024 * 2048)$ ; Note: Convert to Double and Scale by $2^{21} = 1024 * 2048$ $\text{mse\_dbl} = 20 * \log_{10}(\text{mse\_dbl})$ ; Note: Convert to dB Nominal Computed Values for $\text{mse\_dbl}$ are: -22dB to -31dB

**Returns**

Return code. VTSS\_RC\_OK if all Ok VTSS\_RC\_ERROR if and error occurred.

**7.31.5.93 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.

**7.31.5.94 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.

**7.31.5.95 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.

### 7.31.5.96 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.

### 7.31.5.97 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.

## 7.31.5.98 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.

## 7.31.5.99 vtss\_phy\_sd6g\_mac\_serdes\_conf()

```
vtss_rc vtss_phy_sd6g_mac_serdes_conf (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no )
```

Function for Configuring MAC i/f 6G serdes (applicable only on VIPER,ELISE)

## Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Any phy port with the chip

## Returns

VTSS\_RC\_OK when new access can be done - VTSS\_RC\_ERROR if something went wrong and access was never granted.

## 7.32 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"
```

### 7.32.1 Detailed Description

PHY TimeStamping API.

This header file describes PTP/OAM TimeStamping API functions

## 7.33 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.*

### 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 9600

## Enumerations

- enum `vtss_miim_controller_t` { `VTSS_MIIM_CONTROLLER_0` = 0, `VTSS_MIIM_CONTROLLER_1` = 1, `VTSS_MIIM_CONTROLLERS`, `VTSS_MIIM_CONTROLLER_NONE` = -1 }

*MII management controller.*

- enum `vtss_internal_bw_t` { `VTSS_BW_DEFAULT`, `VTSS_BW_1G`, `VTSS_BW_2500M`, `VTSS_BW_UNDEFINED` }

*The internal bandwidth allocated for the port.*

- enum `vtss_port_clause_37_remote_fault_t` { `VTSS_PORT_CLAUSE_37_RF_LINK_OK` = ((0<<1) | (0<<0)), `VTSS_PORT_CLAUSE_37_RF_OFFLINE` = ((1<<1) | (0<<0)), `VTSS_PORT_CLAUSE_37_RF_LINK_FAILURE` = ((0<<1) | (1<<0)), `VTSS_PORT_CLAUSE_37_RF_AUTONEG_ERROR` = ((1<<1) | (1<<0)) }

*Auto-negotiation remote fault type.*

- enum `vtss_port_max_tags_t` { `VTSS_PORT_MAX_TAGS_NONE`, `VTSS_PORT_MAX_TAGS_ONE`, `VTSS_PORT_MAX_TAGS_TWO` }

*VLAN awareness for frame length check.*

- enum `vtss_port_loop_t` { `VTSS_PORT_LOOP_DISABLE`, `VTSS_PORT_LOOP_PCS_HOST` }

*Port loop back configuration.*

- enum `vtss_port_forward_t` { `VTSS_PORT_FORWARD_ENABLED`, `VTSS_PORT_FORWARD_DISABLED`, `VTSS_PORT_FORWARD_INGRESS`, `VTSS_PORT_FORWARD_EGRESS` }

*Port forwarding state.*

## Functions

- `vtss_rc vtss_port_map_set` (const `vtss_inst_t` inst, const `vtss_port_map_t` port\_map[`VTSS_PORT_ARRAY_SIZE`])  
*Set port map.*
- `vtss_rc vtss_port_map_get` (const `vtss_inst_t` inst, `vtss_port_map_t` port\_map[`VTSS_PORT_ARRAY_SIZE`])  
*Get port map.*
- `vtss_rc vtss_port_clause_37_control_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_port_clause_37_control_t` \*const control)  
*Get clause 37 auto-negotiation Control word.*
- `vtss_rc vtss_port_clause_37_control_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_port_clause_37_control_t` \*const control)  
*Set clause 37 auto-negotiation Control word.*
- `vtss_rc vtss_port_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_port_conf_t` \*const conf)  
*Set port configuration. Note: If if\_type in the vtss\_port\_conf\_t/vtss\_port\_interface\_t definition is set to VTSS\_PORT\_INTERFACE\_QSGMII, the ports are mapped together in groups of four. If one of the four ports is used, all four ports in the group must always be configured, but the four ports doesn't need to be configured with the same configuration. This is needed in order to achieve correct comma alignment at the QSGMII interface. Which ports that are mapped together can be found in the chip data-sheet.*
- `vtss_rc vtss_port_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_port_conf_t` \*const conf)  
*Get port setup.*
- `vtss_rc vtss_port_status_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_port_status_t` \*const status)  
*Get port status.*
- `vtss_rc vtss_port_counters_update` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no)  
*Update counters for port.*
- `vtss_rc vtss_port_counters_clear` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no)  
*Clear counters for port.*
- `vtss_rc vtss_port_counters_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_port_counters_t` \*const counters)  
*Get counters for port.*

- `vtss_rc vtss_port_basic_counters_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_basic_counters_t` \*const counters)
 

*Get basic counters for port.*
- `vtss_rc vtss_port_forward_state_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, `vtss_port_forward_t` \*const forward)
 

*Get port forwarding state.*
- `vtss_rc vtss_port_forward_state_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_port_forward_t` forward)
 

*Set port forwarding state.*
- `vtss_rc vtss_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, `u16` value)
 

*Direct MIIM write (bypassing port map)*

### 7.33.1 Detailed Description

Port API.

### 7.33.2 Macro Definition Documentation

#### 7.33.2.1 CHIP\_PORT\_UNUSED

```
#define CHIP_PORT_UNUSED -1
```

Signifies an unused chip port

Definition at line 69 of file vtss\_port\_api.h.

#### 7.33.2.2 VTSS\_FRAME\_GAP\_DEFAULT

```
#define VTSS_FRAME_GAP_DEFAULT 0
```

Default frame gap used

Definition at line 203 of file vtss\_port\_api.h.

### 7.33.2.3 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.

### 7.33.2.4 VTSS\_MAX\_FRAME\_LENGTH\_MAX [1/2]

```
#define VTSS_MAX_FRAME_LENGTH_MAX 10240
```

Maximum frame length supported

Definition at line 219 of file vtss\_port\_api.h.

### 7.33.2.5 VTSS\_MAX\_FRAME\_LENGTH\_MAX [2/2]

```
#define VTSS_MAX_FRAME_LENGTH_MAX 9600
```

Maximum frame length supported

Definition at line 219 of file vtss\_port\_api.h.

## 7.33.3 Enumeration Type Documentation

### 7.33.3.1 vtss\_miim\_controller\_t

```
enum vtss_miim_controller_t
```

MII management controller.

Enumerator

VTSS_MIIM_CONTROLLER_0	MIIM controller 0
VTSS_MIIM_CONTROLLER_1	MIIM controller 1
VTSS_MIIM_CONTROLLERS	Number of MIIM controllers
VTSS_MIIM_CONTROLLER_NONE	Unassigned MIIM controller

Definition at line 42 of file vtss\_port\_api.h.

### 7.33.3.2 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.

### 7.33.3.3 vtss\_port\_clause\_37\_remote\_fault\_t

```
enum vtss_port_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_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.

#### 7.33.3.4 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.

#### 7.33.3.5 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.

#### 7.33.3.6 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.

### 7.33.4 Function Documentation

#### 7.33.4.1 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.

**7.33.4.2 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.

**7.33.4.3 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.

#### 7.33.4.4 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.

#### 7.33.4.5 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.

#### 7.33.4.6 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.

#### 7.33.4.7 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.

#### 7.33.4.8 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.

**7.33.4.9 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.

**7.33.4.10 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.

**7.33.4.11 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.

**7.33.4.12 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.

## 7.33.4.13 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.

## 7.33.4.14 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.

### 7.33.4.15 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.

## 7.34 vtss\_api/include/vtss\_qos\_api.h File Reference

QoS API.

```
#include <vtss/api/types.h>
```

### Data Structures

- struct [vtss\\_qos\\_conf\\_t](#)

*All parameters below are defined per chip.*
- struct [vtss\\_policer\\_t](#)

*Policer.*
- struct [vtss\\_policer\\_ext\\_t](#)

*Policer Extensions.*
- struct [vtss\\_dlb\\_policer\\_conf\\_t](#)

*Dual leaky buckets policer configuration.*
- struct [vtss\\_shaper\\_t](#)

*Shaper.*
- struct [vtss\\_qos\\_port\\_conf\\_t](#)

*QoS setup per port.*
- struct [vtss\\_qce\\_mac\\_t](#)

*QCE MAC information.*
- struct [vtss\\_qce\\_tag\\_t](#)

- *QCE tag information.*
- struct `vtss_qce_frame_etype_t`  
*Frame data for VTSS\_QCE\_TYPEETYPE.*
- struct `vtss_qce_frame_llc_t`  
*Frame data for VTSS\_QCE\_TYPELLC.*
- struct `vtss_qce_frame_snap_t`  
*Frame data for VTSS\_QCE\_TYPESNAP.*
- struct `vtss_qce_frame_ipv4_t`  
*Frame data for VTSS\_QCE\_TYPEIPV4.*
- struct `vtss_qce_frame_ipv6_t`  
*Frame data for VTSS\_QCE\_TYPEIPV6.*
- struct `vtss_qce_key_t`  
*QCE key.*
- struct `vtss_qce_action_t`  
*QCE action.*
- struct `vtss_qce_t`  
*QoS Control Entry.*

## Macros

- `#define VTSS_PORT_POLICERS 1`
- `#define VTSS_PORT_POLICER_CPU_QUEUES 8`
- `#define VTSS_QCL_IDS 1`
- `#define VTSS_QCL_ID_START 0`
- `#define VTSS_QCL_ID_END (VTSS_QCL_ID_START + VTSS_QCL_IDS)`
- `#define VTSS_QCL_ARRAY_SIZE VTSS_QCL_ID_END`
- `#define VTSS_QCE_ID_LAST 0`

## Typedefs

- `typedef u32 vtss_qcl_id_t`  
*QCL ID type.*

## Enumerations

- enum `vtss_tag_remark_mode_t`{ `VTSS_TAG_REMARK_MODE_CLASSIFIED` = 0, `VTSS_TAG_REMARK_MODE_DEFAULT` = 2, `VTSS_TAG_REMARK_MODE_MAPPED` = 3 }  
*Tag Remark Mode.*
- enum `vtss_dscp_mode_t`{ `VTSS_DSCP_MODE_NONE`, `VTSS_DSCP_MODE_ZERO`, `VTSS_DSCP_MODE_SEL`, `VTSS_DSCP_MODE_ALL` }  
*DSCP mode for ingress port.*
- enum `vtss_dscpemode_t` { `VTSS_DSCP_EMODE_DISABLE`, `VTSS_DSCP_EMODE_REMARK`, `VTSS_DSCP_EMODE_REMAP`, `VTSS_DSCP_EMODE_REMAP_DPA` }  
*DSCP mode for egress port.*
- enum `vtss_qce_type_t`{  
  `VTSS_QCE_TYPE_ANY`, `VTSS_QCE_TYPEETYPE`, `VTSS_QCE_TYPE_LLCC`, `VTSS_QCE_TYPE_SNAP`,  
  `VTSS_QCE_TYPE_IPV4`, `VTSS_QCE_TYPE_IPV6` }  
*QoS Control Entry type.*

## Functions

- `vtss_rc vtss_qos_conf_get` (const `vtss_inst_t` inst, `vtss_qos_conf_t` \*const conf)  
*Get QoS setup for switch.*
- `vtss_rc vtss_qos_conf_set` (const `vtss_inst_t` inst, const `vtss_qos_conf_t` \*const conf)  
*Set QoS setup for switch.*
- `vtss_rc vtss_mep_policer_conf_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_prio_t` prio, `vtss_db_policer_conf_t` \*const conf)  
*Get MEP policer configuration.*
- `vtss_rc vtss_mep_policer_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port\_no, const `vtss_prio_t` prio, const `vtss_db_policer_conf_t` \*const conf)  
*Set MEP policer configuration.*
- `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.*

### 7.34.1 Detailed Description

QoS API.

This header file describes Quality of Service functions

### 7.34.2 Macro Definition Documentation

#### 7.34.2.1 VTSS\_PORT\_POLICERS

```
#define VTSS_PORT_POLICERS 1
```

Number of port policers

Definition at line 179 of file vtss\_qos\_api.h.

### 7.34.2.2 VTSS\_PORT\_POLICER\_CPU\_QUEUES

```
#define VTSS_PORT_POLICER_CPU_QUEUES 8
```

Number of cpu queues pr port policer

Definition at line 193 of file vtss\_qos\_api.h.

### 7.34.2.3 VTSS\_QCL\_IDS

```
#define VTSS_QCL_IDS 1
```

Number of QCLs

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

### 7.34.2.4 VTSS\_QCL\_ID\_START

```
#define VTSS_QCL_ID_START 0
```

QCL ID start number

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

### 7.34.2.5 VTSS\_QCL\_ID\_END

```
#define VTSS_QCL_ID_END (VTSS_QCL_ID_START + VTSS_QCL_IDS)
```

QCL ID end number

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

### 7.34.2.6 VTSS\_QCL\_ARRAY\_SIZE

```
#define VTSS_QCL_ARRAY_SIZE VTSS_QCL_ID_END
```

QCL ID array size

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

### 7.34.2.7 VTSS\_QCE\_ID\_LAST

```
#define VTSS_QCE_ID_LAST 0
```

Special value used to add last in list

Definition at line 457 of file vtss\_qos\_api.h.

## 7.34.3 Enumeration Type Documentation

### 7.34.3.1 vtss\_tag\_remark\_mode\_t

```
enum vtss_tag_remark_mode_t
```

Tag Remark Mode.

#### Enumerator

VTSS_TAG_REMARK_MODE_CLASSIFIED	Use classified PCP/DEI values
VTSS_TAG_REMARK_MODE_DEFAULT	Use default (configured) PCP/DEI values
VTSS_TAG_REMARK_MODE_MAPPED	Use mapped versions of classified QOS class and DP level

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

### 7.34.3.2 vtss\_dscp\_mode\_t

```
enum vtss_dscp_mode_t
```

DSCP mode for ingress port.

#### Enumerator

VTSS_DSCP_MODE_NONE	DSCP not remarked
VTSS_DSCP_MODE_ZERO	DSCP value zero remarked
VTSS_DSCP_MODE_SEL	DSCP values selected above (dscp_remark) are remarked
VTSS_DSCP_MODE_ALL	DSCP remarked for all values

Definition at line 322 of file vtss\_qos\_api.h.

### 7.34.3.3 vtss\_dscp\_emode\_t

```
enum vtss_dscp_emode_t
```

DSCP mode for egress port.

#### Enumerator

VTSS_DSCP_EMODE_DISABLE	DSCP not remarked
VTSS_DSCP_EMODE_REMARK	DSCP remarked with DSCP value from analyzer
VTSS_DSCP_EMODE_REMAP	DSCP remarked with DSCP value from analyzer remapped through global remap table
VTSS_DSCP_EMODE_REMAP_DPA	DSCP remarked with DSCP value from analyzer remapped through global remap dp aware tables

Definition at line 333 of file vtss\_qos\_api.h.

#### 7.34.3.4 vtss\_qce\_type\_t

```
enum vtss_qce_type_t
```

QoS Control Entry type.

#### Enumerator

VTSS_QCE_TYPE_ANY	Any frame type
VTSS_QCE_TYPE_ETYPE	Ethernet Type
VTSS_QCE_TYPE_LLC	LLC
VTSS_QCE_TYPE_SNAP	SNAP
VTSS_QCE_TYPE_IPV4	IPv4
VTSS_QCE_TYPE_IPV6	IPv6

Definition at line 460 of file vtss\_qos\_api.h.

### 7.34.4 Function Documentation

#### 7.34.4.1 vtss\_qos\_conf\_get()

```
vtss_rc vtss_qos_conf_get (
    const vtss_inst_t inst,
    vtss_qos_conf_t *const conf )
```

Get QoS setup for switch.

#### Parameters

<i>inst</i>	[IN] Target instance reference.
<i>conf</i>	[OUT] QoS setup structure.

**Returns**

Return code.

**7.34.4.2 vtss\_qos\_conf\_set()**

```
vtss_rc vtss_qos_conf_set (
    const vtss_inst_t inst,
    const vtss_qos_conf_t *const conf )
```

Set QoS setup for switch.

**Parameters**

<i>inst</i>	[IN] Target instance reference.
<i>conf</i>	[IN] QoS setup structure.

**Returns**

Return code.

**7.34.4.3 vtss\_mep\_policer\_conf\_get()**

```
vtss_rc vtss_mep_policer_conf_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_prio_t prio,
    vtss_dlb_policer_conf_t *const conf )
```

Get MEP policer configuration.

**Parameters**

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Ingress port number.
<i>prio</i>	[IN] Selected priority (QoS class).
<i>conf</i>	[OUT] Policer configuration.

**Returns**

Return code.

#### 7.34.4.4 vtss\_mep\_policer\_conf\_set()

```
vtss_rc vtss_mep_policer_conf_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_prio_t prio,
    const vtss_dlb_policer_conf_t *const conf )
```

Set MEP policer configuration.

##### Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Ingress port number.
<i>prio</i>	[IN] Selected priority (QoS class).
<i>conf</i>	[IN] Policer configuration.

##### Returns

Return code.

#### 7.34.4.5 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.

#### 7.34.4.6 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.

**7.34.4.7 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.

**7.34.4.8 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.

**7.34.4.9 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.

**7.35 vtss\_api/include/vtss\_rab\_api.h File Reference**

RAB API.

```
#include <vtss/api/types.h>
```

**7.35.1 Detailed Description**

RAB API.

**7.36 vtss\_api/include/vtss\_security\_api.h File Reference**

Security API.

```
#include <vtss/api/types.h>
```

## Data Structures

- struct `vtss_acl_policer_conf_t`  
*ACL policer configuration.*
- struct `vtss_acl_action_t`  
*ACL Action.*
- struct `vtss_acl_port_conf_t`  
*ACL port configuration.*
- struct `vtss_ace_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_AND_TWO_STEP, VTSS_ACL_PTP_ACTION_TWO_STEP }`  
*ACL PTP action.*
- `enum vtss_ace_type_t { VTSS_ACE_TYPE_ANY, VTSS_ACE_TYPEETYPE, VTSS_ACE_TYPE_LLC, 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.*

### 7.36.1 Detailed Description

Security API.

This header file describes security functions

### 7.36.2 Macro Definition Documentation

#### 7.36.2.1 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.

### 7.36.3 Enumeration Type Documentation

#### 7.36.3.1 vtss\_auth\_state\_t

```
enum vtss_auth_state_t
```

Authentication state.

Enumerator

<code>VTSS_AUTH_STATE_NONE</code>	Not authenticated
<code>VTSS_AUTH_STATE_EGRESS</code>	Authenticated in egress direction
<code>VTSS_AUTH_STATE_BOTH</code>	Authenticated in both directions

Definition at line 59 of file vtss\_security\_api.h.

### 7.36.3.2 vtss\_acl\_port\_action\_t

enum `vtss_acl_port_action_t`

ACL port action.

Enumerator

<code>VTSS_ACL_PORT_ACTION_NONE</code>	No action from port list
<code>VTSS_ACL_PORT_ACTION_FILTER</code>	Port list filter is used
<code>VTSS_ACL_PORT_ACTION_REDIR</code>	Port list redirect is used

Definition at line 194 of file vtss\_security\_api.h.

### 7.36.3.3 vtss\_acl\_ptp\_action\_t

enum `vtss_acl_ptp_action_t`

ACL PTP action.

Enumerator

<code>VTSS_ACL_PTP_ACTION_NONE</code>	No PTP action
<code>VTSS_ACL_PTP_ACTION_ONE_STEP</code>	PTP one-step time-stamping
<code>VTSS_ACL_PTP_ACTION_ONE_AND_TWO_STEP</code>	PTP one-step and two-step time-stamping
<code>VTSS_ACL_PTP_ACTION_TWO_STEP</code>	PTP two-step time-stamping

Definition at line 202 of file vtss\_security\_api.h.

### 7.36.3.4 vtss\_ace\_type\_t

enum `vtss_ace_type_t`

ACE frame type.

Enumerator

<code>VTSS_ACE_TYPE_ANY</code>	Any frame type
<code>VTSS_ACE_TYPE_ETYPE</code>	Ethernet Type
<code>VTSS_ACE_TYPE_LLC</code>	LLC
<code>VTSS_ACE_TYPE_SNAP</code>	SNAP
<code>VTSS_ACE_TYPE_ARP</code>	ARP/RARP
<code>VTSS_ACE_TYPE_IPV4</code>	IPv4
<code>VTSS_ACE_TYPE_IPV6</code>	IPv6

Definition at line 338 of file vtss\_security\_api.h.

#### 7.36.3.5 vtss\_ace\_bit\_t

enum `vtss_ace_bit_t`

ACE 1 bit.

Enumerator

VTSS_ACE_BIT_ANY	Value 0 or 1
VTSS_ACE_BIT_0	Value 0
VTSS_ACE_BIT_1	Value 1

Definition at line 355 of file vtss\_security\_api.h.

### 7.36.4 Function Documentation

#### 7.36.4.1 vtss\_auth\_port\_state\_get()

```
vtss_rc vtss_auth_port_state_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_auth_state_t *const state )
```

Get 802.1X Authentication state for a port.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>state</i>	[OUT] Authentication state.

Returns

Return code.

#### 7.36.4.2 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.

#### 7.36.4.3 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.

#### 7.36.4.4 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.

**7.36.4.5 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.

**7.36.4.6 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.

**7.36.4.7 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.

**7.36.4.8 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.

**7.36.4.9 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.

### 7.36.4.10 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.

### 7.36.4.11 vtss\_ace\_del()

```
vtss_rc vtss_ace_del (
    const vtss_inst_t inst,
    const vtss_ace_id_t ace_id )
```

Delete ACE.

#### Parameters

<i>inst</i>	[IN] Target instance reference.
<i>ace_id</i>	[IN] ACE ID.

**Returns**

Return code.

**7.36.4.12 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.

**7.36.4.13 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.

**7.37 vtss\_api/include/vtss\_sfi4\_api.h File Reference****SFI4 API.**

```
#include <vtss/api/types.h>
```

### 7.37.1 Detailed Description

SFI4 API.

## 7.38 vtss\_api/include/vtss\_sync\_api.h File Reference

Synchronization API.

```
#include "vtss/api/types.h"
```

### Data Structures

- struct `vtss_sync_clock_out_t`  
*Struct containing configuration for a recovered clock output port.*
- struct `vtss_sync_clock_in_t`  
*Struct containing configuration selecting the recovered input clock port, to be delivered to a selected output clock port.*

### Macros

- #define `VTSS_SYNC_CLK_A` 0
- #define `VTSS_SYNC_CLK_B` 1
- #define `VTSS_SYNC_CLK_MAX` 2

### Typedefs

- typedef `u32 vtss_sync_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.*

### 7.38.1 Detailed Description

Synchronization API.

This header file describes synchronization functions

### 7.38.2 Macro Definition Documentation

#### 7.38.2.1 VTSS\_SYNCE\_CLK\_A

```
#define VTSS_SYNCE_CLK_A 0
```

Clock A output port

Definition at line 56 of file vtss\_sync\_api.h.

#### 7.38.2.2 VTSS\_SYNCE\_CLK\_B

```
#define VTSS_SYNCE_CLK_B 1
```

Clock B output port

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

#### 7.38.2.3 VTSS\_SYNCE\_CLK\_MAX

```
#define VTSS_SYNCE_CLK_MAX 2
```

Number of recovered clock outputs

Definition at line 61 of file vtss\_sync\_api.h.

### 7.38.3 Enumeration Type Documentation

#### 7.38.3.1 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.

#### 7.38.4 Function Documentation

##### 7.38.4.1 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.

##### 7.38.4.2 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.

**7.38.4.3 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.

**7.38.4.4 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.

**7.39 vtss\_api/include/vtss\_tfi5\_api.h File Reference**

TFI5 API.

```
#include <vtss/api/types.h>
```

### 7.39.1 Detailed Description

TFI5 API.

## 7.40 vtss\_api/include/vtss\_ts\_api.h File Reference

TimeStamping API.

```
#include <vtss/api/types.h>
```

### Data Structures

- struct [vtss\\_ts\\_ext\\_clock\\_mode\\_t](#)  
*external clock output configuration.*
- struct [vtss\\_ts\\_operation\\_mode\\_t](#)  
*Timestamp operation.*
- struct [vtss\\_ts\\_internal\\_mode\\_t](#)  
*Hardware timestamping format mode for internal ports.*
- struct [vtss\\_ts\\_id\\_t](#)  
*Timestamp identifier.*
- struct [vtss\\_ts\\_timestamp\\_t](#)  
*Timestamp structure.*
- struct [vtss\\_ts\\_timestamp\\_alloc\\_t](#)  
*Timestamp allocation.*

### Macros

- #define [VTSS\\_HW\\_TIME\\_CNT\\_PR\\_SEC](#) 250000000 /\* L26 counts clock cycles instead of ns \*/  
*Number of clock cycle counts pr sec.*
- #define [VTSS\\_HW\\_TIME\\_NSEC\\_PR\\_CNT](#) 4  
*Number of nanoseconds pr clock count.*
- #define [VTSS\\_HW\\_TIME\\_WRAP\\_LIMIT](#) 0 /\* time counter wrap around limit+1 (=0 if wrap at 0xffffffff) \*/  
*Caracal nanosecond time counter wrap around value (Caracal time counter wraps when 0xffffffff is reached).*
- #define [VTSS\\_HW\\_TIME\\_MIN\\_ADJ\\_RATE](#) 40 /\* 4 ppb \*/  
*Jaguar/Luton26 minimum adjustment rate in units of 0,1 ppb.*
- #define [VTSS\\_HW\\_TIME\\_MAX\\_FINE\\_ADJ](#) 25  
*This is the max time offset adjustment that os done without setting ports in disabled state.*

## Typedefs

- `typedef struct vtss_ts_ext_clock_mode_t vtss_ts_ext_clock_mode_t`  
*external clock output configuration.*
- `typedef struct vtss_ts_operation_mode_t vtss_ts_operation_mode_t`  
*Timestamp operation.*
- `typedef struct vtss_ts_internal_mode_t vtss_ts_internal_mode_t`  
*Hardware timestamping format mode for internal ports.*
- `typedef struct vtss_ts_id_t vtss_ts_id_t`  
*Timestamp identifier.*
- `typedef struct vtss_ts_timestamp_t vtss_ts_timestamp_t`  
*Timestamp structure.*
- `typedef struct vtss_ts_timestamp_alloc_t vtss_ts_timestamp_alloc_t`  
*Timestamp allocation.*

## Enumerations

- `enum vtss_ts_ext_clock_one_pps_mode_t {  
 TS_EXT_CLOCK_MODE_ONE_PPS_DISABLE, TS_EXT_CLOCK_MODE_ONE_PPS_OUTPUT, TS_EXT_CLOCK_MODE_ONE_PPS_INPUT, TS_EXT_CLOCK_MODE_ONE_PPS_OUTPUT_INPUT,  
 TS_EXT_CLOCK_MODE_MAX }`  
*parameter for setting the external clock mode.*
- `enum vtss_ts_mode_t { TS_MODE_NONE, TS_MODE_EXTERNAL, TS_MODE_INTERNAL, TX_MODE_MAX }`  
*parameter for setting the timestamp operating mode*
- `enum vtss_ts_internal_fmt_t {  
 TS_INTERNAL_FMT_NONE, TS_INTERNAL_FMT_RESERVED_LEN_30BIT, TS_INTERNAL_FMT_RESERVED_LEN_32BIT, TS_INTERNAL_FMT_SUB_ADD_LEN_44BIT_CF62,  
 TS_INTERNAL_FMT_RESERVED_LEN_48BIT_CF_3_0, TS_INTERNAL_FMT_RESERVED_LEN_48BIT_CF_0, TS_INTERNAL_FMT_RESERVED_LEN_48BIT_CF, TX_INTERNAL_FMT_MAX }`  
*parameter for setting the internal timestamp format*

## Functions

- `vtss_rc vtss_ts_timeofday_set (const vtss_inst_t inst, const vtss_timestamp_t *const ts)`  
*Set the current time in a Timestamp format.*
- `vtss_rc vtss_ts_timeofday_set_delta (const vtss_inst_t inst, const vtss_timestamp_t *ts, BOOL negative)`  
*Set delta the current time in a Timestamp format.*
- `vtss_rc vtss_ts_timeofday_offset_set (const vtss_inst_t inst, const i32 offset)`  
*Subtract offset from the current time.*
- `vtss_rc vtss_ts_adjtimer_one_sec (const vtss_inst_t inst, BOOL *const ongoing_adjustment)`  
*Do the one sec administration in the Timestamp function.*
- `vtss_rc vtss_ts_ongoing_adjustment (const vtss_inst_t inst, BOOL *const ongoing_adjustment)`  
*Check if the clock adjustment is ongoing.*
- `vtss_rc vtss_ts_timeofday_get (const vtss_inst_t inst, vtss_timestamp_t *const ts, u32 *const tc)`  
*Get the current time in a Timestamp format, and the corresponding time counter.*
- `vtss_rc vtss_ts_timeofday_next_pps_get (const vtss_inst_t inst, vtss_timestamp_t *const ts)`  
*Get the time at the next 1PPS pulse edge in a Timestamp format.*
- `vtss_rc vtss_ts_adjtimer_set (const vtss_inst_t inst, const i32 adj)`  
*Adjust the clock timer ratio.*
- `vtss_rc vtss_ts_adjtimer_get (const vtss_inst_t inst, i32 *const adj)`

- `vtss_rc vtss_ts_freq_offset_get` (const `vtss_inst_t` inst, `i32` \*const adj)
 

*get the clock timer ratio.*
- `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 timestamp 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, const alloc_parm_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)*

#### 7.40.1 Detailed Description

TimeStamping API.

This header file describes PTP/OAM TimeStamping API functions and associated types.

#### 7.40.2 Function Documentation

##### 7.40.2.1 vtss\_ts\_timeofday\_set()

```
vtss_rc vtss_ts_timeofday_set (
    const vtss_inst_t inst,
    const vtss_timestamp_t *const ts )
```

Set the current time in a Timestamp format.

##### Parameters

<code>inst</code>	[IN] handle to an API instance.
<code>ts</code>	[IN] pointer to a TimeStamp structure.

##### Returns

Return code.

##### 7.40.2.2 vtss\_ts\_timeofday\_set\_delta()

```
vtss_rc vtss_ts_timeofday_set_delta (
    const vtss_inst_t inst,
    const vtss_timestamp_t * ts,
    BOOL negative )
```

Set delta the current time in a Timestamp format.

**Parameters**

<i>inst</i>	[IN] handle to an API instance.
<i>ts</i>	[IN] pointer to a TimeStamp structure.
<i>negative</i>	[IN] True if ts is subtracted from current time, else ts is added.

**Returns**

Return code.

**7.40.2.3 vtss\_ts\_timeofday\_offset\_set()**

```
vtss_rc vtss_ts_timeofday_offset_set (
    const vtss_inst_t inst,
    const i32 offset )
```

Subtract offset from the current time.

**Parameters**

<i>inst</i>	[IN] handle to an API instance.
<i>offset</i>	[IN] offset in ns.

**Returns**

Return code.

**7.40.2.4 vtss\_ts\_adjtimer\_one\_sec()**

```
vtss_rc vtss_ts_adjtimer_one_sec (
    const vtss_inst_t inst,
    BOOL *const ongoing_adjustment )
```

Do the one sec administration in the Timestamp function.

**Parameters**

<i>inst</i>	[IN] handle to an API instance.
<i>ongoing_adjustment</i>	[OUT] True if clock adjustment is ongoing

**Returns**

Return code.

Jr1 : Maintains the clock setting process Caracal: Maintains the clock setting process Serval1: Maintains the clock setting process JR2 : it must only be called when the PPS output pin is low, therefore it shall be called at least 200 microseconds after the 1PPS interrupt

#### 7.40.2.5 vtss\_ts\_ongoing\_adjustment()

```
vtss_rc vtss_ts_ongoing_adjustment (
    const vtss_inst_t inst,
    BOOL *const ongoing_adjustment )
```

Check if the clock adjustment is ongoing.

##### Parameters

<i>inst</i>	[IN] handle to an API instance.
<i>ongoing_adjustment</i>	[OUT] True if clock adjustment is ongoing

##### Returns

Return code.

#### 7.40.2.6 vtss\_ts\_timeofday\_get()

```
vtss_rc vtss_ts_timeofday_get (
    const vtss_inst_t inst,
    vtss_timestamp_t *const ts,
    u32 *const tc )
```

Get the current time in a Timestamp format, and the corresponding time counter.

##### Parameters

<i>inst</i>	[IN] handle to an API instance
<i>ts</i>	[OUT] pointer to a TimeStamp structure
<i>tc</i>	[OUT] pointer to a time counter (internal hw format) Jaguar: tc = nanoseconds/nanosec_pr_clock_cycle (0..249999999) Caracal:tc = free running clock cycle counter Serval: tc = (nanoseconds + seconds*10**9) mod 2**32

##### Returns

Return code.

#### 7.40.2.7 vtss\_ts\_timeofday\_next\_pps\_get()

```
vtss_rc vtss_ts_timeofday_next_pps_get (
    const vtss_inst_t inst,
    vtss_timestamp_t *const ts )
```

Get the time at the next 1PPS pulse edge in a Timestamp format.

#### Parameters

<i>inst</i>	[IN] handle to an API instance
<i>ts</i>	[OUT] pointer to a TimeStamp structure

#### Returns

Return code.

### 7.40.2.8 vtss\_ts\_adjtimer\_set()

```
vtss_rc vtss_ts_adjtimer_set (
    const vtss_inst_t inst,
    const i32 adj )
```

Adjust the clock timer ratio.

#### Parameters

<i>inst</i>	[IN] handle to an API instance.
<i>adj</i>	[IN] Clock ratio frequency offset in units of 0,1 ppb (parts pr billion). ratio > 0 => clock runs faster

#### Returns

Return code.

### 7.40.2.9 vtss\_ts\_adjtimer\_get()

```
vtss_rc vtss_ts_adjtimer_get (
    const vtss_inst_t inst,
    i32 *const adj )
```

get the clock timer ratio.

#### Parameters

<i>inst</i>	[IN] handle to an API instance.
<i>adj</i>	[OUT] Clock ratio frequency offset in ppb (parts pr billion). ratio > 0 => clock runs faster

#### Returns

Return code.

#### 7.40.2.10 vtss\_ts\_freq\_offset\_get()

```
vtss_rc vtss_ts_freq_offset_get (
    const vtss_inst_t inst,
    i32 *const adj )
```

get the clock internal timer frequency offset, compared to external clock input.

##### Parameters

<i>inst</i>	[IN] handle to an API instance.
<i>adj</i>	[OUT] Clock ratio frequency offset in ppb (parts pr billion). ratio > 0 => internal clock runs faster than external clock

##### Returns

Return code.

#### 7.40.2.11 vtss\_ts\_external\_clock\_mode\_get()

```
vtss_rc vtss_ts_external_clock_mode_get (
    const vtss_inst_t inst,
    vtss_ts_ext_clock_mode_t *const ext_clock_mode )
```

Get the external clock mode. The mode depends on the hardware capability, it may be: Enable/disable external synch pulse Set clock output frequency.

##### Parameters

<i>inst</i>	[IN] handle to an API instance
<i>ext_clock_mode</i>	[OUT] external clock mode.

##### Returns

Return code.

#### 7.40.2.12 vtss\_ts\_external\_clock\_mode\_set()

```
vtss_rc vtss_ts_external_clock_mode_set (
    const vtss_inst_t inst,
    const vtss_ts_ext_clock_mode_t *const ext_clock_mode )
```

Set the external clock mode. The mode depends on the hardware capability, it may be: Enable/disable external synch pulse Set clock output frequency.

**Parameters**

<i>inst</i>	[IN] handle to an API instance
<i>ext_clock_mode</i>	[IN] external clock mode.

**Returns**

Return code.

**7.40.2.13 vtss\_ts\_external\_clock\_saved\_get()**

```
vtss_rc vtss_ts_external_clock_saved_get (
    const vtss_inst_t inst,
    u32 *const saved )
```

Get the latest saved time counter in nanosec.

**Parameters**

<i>inst</i>	[IN] handle to an API instance
<i>saved</i>	[OUT] latest saved value. [0..999.999.999]

**Returns**

Return code.

**7.40.2.14 vtss\_ts\_ingress\_latency\_set()**

```
vtss_rc vtss_ts_ingress_latency_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_timeinterval_t *const ingress_latency )
```

Set the ingress latency.

**Parameters**

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>ingress_latency</i>	[IN] pointer to ingress latency

**Returns**

Return code.

#### 7.40.2.15 vtss\_ts\_ingress\_latency\_get()

```
vtss_rc vtss_ts_ingress_latency_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_timeinterval_t *const ingress_latency )
```

Get the ingress latency.

##### Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>ingress_latency</i>	[OUT] pointer to ingress_latency

##### Returns

Return code.

#### 7.40.2.16 vtss\_ts\_p2p\_delay\_set()

```
vtss_rc vtss_ts_p2p_delay_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_timeinterval_t *const p2p_delay )
```

Set the P2P delay.

##### Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>p2p_delay</i>	[IN] peer-2-peer delay (measured)

##### Returns

Return code.

#### 7.40.2.17 vtss\_ts\_p2p\_delay\_get()

```
vtss_rc vtss_ts_p2p_delay_get (
    const vtss_inst_t inst,
```

```
const vtss_port_no_t port_no,
      vtss_timeinterval_t *const p2p_delay )
```

Get the P2P delay.

#### Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>p2p_delay</i>	[OUT] pointer to peer-2-peer delay

#### Returns

Return code.

### 7.40.2.18 vtss\_ts\_egress\_latency\_set()

```
vtss_rc vtss_ts_egress_latency_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_timeinterval_t *const egress_latency )
```

Set the egress latency.

#### Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>egress_latency</i>	[IN] egress latency

#### Returns

Return code.

### 7.40.2.19 vtss\_ts\_egress\_latency\_get()

```
vtss_rc vtss_ts_egress_latency_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_timeinterval_t *const egress_latency )
```

Get the egress latency.

#### Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>egress_latency</i>	[OUT] pointer to egress latency

**Returns**

Return code.

**7.40.2.20 vtss\_ts\_delay\_asymmetry\_set()**

```
vtss_rc vtss_ts_delay_asymmetry_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_timeinterval_t *const delay_asymmetry )
```

Set the delay asymmetry.

**Parameters**

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>delay_asymmetry</i>	[IN] delay asymmetry

**Returns**

Return code.

**7.40.2.21 vtss\_ts\_delay\_asymmetry\_get()**

```
vtss_rc vtss_ts_delay_asymmetry_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_timeinterval_t *const delay_asymmetry )
```

Get the delay asymmetry.

**Parameters**

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>delay_asymmetry</i>	[OUT] pointer to delay asymmetry

**Returns**

Return code.

#### 7.40.2.22 vtss\_ts\_operation\_mode\_set()

```
vtss_rc vtss_ts_operation_mode_set (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    const vtss_ts_operation_mode_t *const mode )
```

Set the timestamping operation mode for a port.

##### Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>mode</i>	[IN] pointer to a struct holding the operation mode

##### Returns

Return code.

Serval: Used to set backplane (INTERNAL) mode/normal(EXTERNAL) mode Other : Not used

#### 7.40.2.23 vtss\_ts\_operation\_mode\_get()

```
vtss_rc vtss_ts_operation_mode_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_ts_operation_mode_t *const mode )
```

Get the timestamping operation mode for a port.

##### Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>mode</i>	[OUT] pointer to a struct holding the operation mode

##### Returns

Return code.

#### 7.40.2.24 vtss\_ts\_internal\_mode\_set()

```
vtss_rc vtss_ts_internal_mode_set (
    const vtss_inst_t inst,
    const vtss_ts_internal_mode_t *const mode )
```

Set the internal timestamping mode.

**Parameters**

<i>inst</i>	[IN] handle to an API instance
<i>mode</i>	[IN] pointer to a struct holding the operation mode

**Returns**

Return code.

Serval: Used to set INTERNAL mode timestamping format Other : Not used

**7.40.2.25 vtss\_ts\_internal\_mode\_get()**

```
vtss_rc vtss_ts_internal_mode_get (
    const vtss_inst_t inst,
    vtss_ts_internal_mode_t *const mode )
```

Get the internal timestamping mode.

**Parameters**

<i>inst</i>	[IN] handle to an API instance
<i>mode</i>	[OUT] pointer to a struct holding the operation mode

**Returns**

Return code.

**7.40.2.26 vtss\_tx\_timestamp\_update()**

```
vtss_rc vtss_tx_timestamp_update (
    const vtss_inst_t inst )
```

Update the internal timestamp table, from HW.

**Parameters**

<i>inst</i>	[IN] handle to an API instance
-------------	--------------------------------

**Returns**

Return code.

#### 7.40.2.27 vtss\_rx\_timestamp\_get()

```
vtss_rc vtss_rx_timestamp_get (
    const vtss_inst_t inst,
    const vtss_ts_id_t *const ts_id,
    vtss_ts_timestamp_t *const ts )
```

Get the rx FIFO timestamp for a {timestampId}.

##### Parameters

<i>inst</i>	[IN] handle to an API instance
<i>ts_id</i>	[IN] timestamp id
<i>ts</i>	[OUT] pointer to a struct holding the fifo timestamp

##### Returns

Return code.

#### 7.40.2.28 vtss\_rx\_timestamp\_id\_release()

```
vtss_rc vtss_rx_timestamp_id_release (
    const vtss_inst_t inst,
    const vtss_ts_id_t *const ts_id )
```

Release the FIFO rx timestamp id.

##### Parameters

<i>inst</i>	[IN] handle to an API instance
<i>ts_id</i>	[IN] timestamp id

##### Returns

Return code.

#### 7.40.2.29 vtss\_rx\_master\_timestamp\_get()

```
vtss_rc vtss_rx_master_timestamp_get (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no,
    vtss_ts_timestamp_t *const ts )
```

Get rx timestamp from a port (convert from slave time to the master time)

**Parameters**

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number
<i>ts</i>	[IN/OUT] pointer to a struct holding the timestamp

**Returns**

Return code.

**7.40.2.30 vtss\_tx\_timestamp\_idx\_alloc()**

```
vtss_rc vtss_tx_timestamp_idx_alloc (
    const vtss_inst_t inst,
    const vtss_ts_timestamp_alloc_t *const alloc_parm,
    vtss_ts_id_t *const ts_id )
```

Allocate a timestamp id for a two step transmission.

**Parameters**

<i>inst</i>	[IN] handle to an API instance
<i>alloc_parm</i>	[IN] pointer allocation parameters
<i>ts_id</i>	[OUT] timestamp id

**Returns**

Return code.

**7.40.2.31 vtss\_timestamp\_age()**

```
vtss_rc vtss_timestamp_age (
    const vtss_inst_t inst )
```

Age the FIFO timestamps.

**Parameters**

<i>inst</i>	[IN] handle to an API instance
-------------	--------------------------------

**Returns**

Return code.

### 7.40.2.32 vtss\_ts\_status\_change()

```
vtss_rc vtss_ts_status_change (
    const vtss_inst_t inst,
    const vtss_port_no_t port_no )
```

Signal port status change (used to detect and compensate for the internal ingress and egress latencies)

#### Parameters

<i>inst</i>	[IN] handle to an API instance
<i>port_no</i>	[IN] port number

#### Returns

Return code.

## 7.41 vtss\_api/include/vtss\_upi\_api.h File Reference

Define UPI API interface.

```
#include <vtss/api/types.h>
```

### 7.41.1 Detailed Description

Define UPI API interface.

## 7.42 vtss\_api/include/vtss\_wis\_api.h File Reference

eWIS layer API

```
#include <vtss/api/types.h>
```

### 7.42.1 Detailed Description

eWIS layer API

## 7.43 vtss\_api/include/vtss\_xaui\_api.h File Reference

XAUI API.

```
#include <vtss/api/types.h>
```

### 7.43.1 Detailed Description

XAUI API.

## 7.44 vtss\_api/include/vtss\_xfi\_api.h File Reference

XFI API.

```
#include <vtss/api/types.h>
```

### 7.44.1 Detailed Description

XFI API.



# Index

acknowledge  
    vtss\_port\_clause\_37\_adv\_t, 199

acl\_hit  
    vtss\_packet\_rx\_info\_t, 137

acl\_idx  
    vtss\_packet\_rx\_info\_t, 137

action  
    vtss\_ace\_t, 47  
    vtss\_acl\_port\_conf\_t, 55  
    vtss\_ece\_t, 78  
    vtss\_mce\_t, 120  
    vtss\_qce\_t, 244  
    vtss\_vce\_t, 303

active  
    vtss\_irq\_status\_t, 106

addr  
    vtss\_ip\_addr\_t, 99  
    vtss\_ipv6\_t, 104  
    vtss\_mac\_t, 113  
    vtss\_wol\_mac\_addr\_t, 314

address  
    vtss\_ip\_network\_t, 100  
    vtss\_ipv4\_network\_t, 101  
    vtss\_ipv6\_network\_t, 103

adv\_dis  
    port\_custom\_conf\_t, 25

advertisement  
    vtss\_port\_clause\_37\_control\_t, 201

aged  
    vtss\_mac\_table\_entry\_t, 114  
    vtss\_mac\_table\_status\_t, 116

aggr\_code  
    vtss\_packet\_tx\_info\_t, 156

aggr\_rx\_disable  
    vtss\_packet\_frame\_info\_t, 126  
    vtss\_packet\_port\_info\_t, 128

aggr\_tx\_disable  
    vtss\_packet\_frame\_info\_t, 126  
    vtss\_packet\_port\_info\_t, 128

aneg  
    vtss\_phy\_conf\_t, 168  
    vtss\_port\_status\_t, 230

aneg\_complete  
    vtss\_port\_sgmii\_aneg\_t, 228  
    vtss\_port\_status\_t, 229

aneg\_enable  
    vtss\_phy\_tbi\_conf\_t, 189

aneg\_pd\_detect  
    vtss\_phy\_media\_serdes\_pcs\_ctrl\_t, 180

aneg\_restart  
    vtss\_phy\_mac\_serdes\_pcs\_ctrl\_t, 177

arp  
    vtss\_ace\_frame\_arp\_t, 28  
    vtss\_ace\_t, 48

arrived\_tagged  
    vtss\_packet\_rx\_header\_t, 131

asymmetric\_pause  
    vtss\_phy\_aneg\_t, 164  
    vtss\_port\_clause\_37\_adv\_t, 199

automatic  
    vtss\_learn\_mode\_t, 112

autoneg  
    port\_custom\_conf\_t, 24

BOOL  
    types.h, 374

base\_port\_no  
    vtss\_phy\_type\_t, 191

bit\_count  
    vtss\_sgpi\_conf\_t, 264

bit\_rate  
    vtss\_acl\_policer\_conf\_t, 54

bit\_rate\_enable  
    vtss\_acl\_policer\_conf\_t, 54

bmode  
    vtss\_sgpi\_conf\_t, 264

bpdu\_cpu\_only  
    vtss\_packet\_rx\_reg\_t, 151

bpdu\_queue  
    vtss\_packet\_rx\_queue\_map\_t, 148

bpdu\_reg  
    vtss\_packet\_rx\_port\_conf\_t, 146

bridge  
    vtss\_port\_counters\_t, 206

bytes  
    vtss\_counter\_pair\_t, 62

CHIP\_PORT\_UNUSED  
    vtss\_port\_api.h, 628

cal\_done  
    vtss\_lcpll\_status\_t, 110

cal\_error  
    vtss\_lcpll\_status\_t, 110  
    vtss\_rcpll\_status\_t, 257

cal\_not\_done  
    vtss\_rcpll\_status\_t, 257

cb  
    vtss\_ts\_timestamp\_alloc\_t, 278

cbs

vtss\_dlb\_policer\_conf\_t, 67  
cf  
vtss\_dlb\_policer\_conf\_t, 66  
cfg  
vtss\_phy\_conf\_1g\_t, 166  
cfi  
vtss\_ace\_vlan\_t, 50  
vtss\_tci\_t, 270  
channel\_id  
vtss\_phy\_type\_t, 191  
chip\_no  
vtss\_debug\_info\_t, 63  
vtss\_debug\_lock\_t, 65  
vtss\_packet\_rx\_meta\_t, 142  
vtss\_port\_map\_t, 217  
chip\_port  
vtss\_port\_map\_t, 216  
cir  
vtss\_dlb\_policer\_conf\_t, 67  
clear  
vtss\_debug\_info\_t, 64  
connector\_enable  
vtss\_phy\_loopback\_t, 175  
context  
vtss\_ts\_timestamp\_alloc\_t, 278  
vtss\_ts\_timestamp\_t, 279  
copper  
vtss\_port\_status\_t, 230  
copy\_to\_cpu  
vtss\_mac\_table\_entry\_t, 114  
cos  
vtss\_packet\_rx\_info\_t, 136  
vtss\_packet\_tx\_info\_t, 156  
cpu  
vtss\_acl\_action\_t, 51  
vtss\_learn\_mode\_t, 112  
cpu\_once  
vtss\_acl\_action\_t, 51  
cpu\_queue  
vtss\_acl\_action\_t, 51  
vtss\_mac\_table\_entry\_t, 114  
cs\_wait\_ns  
vtss\_pi\_conf\_t, 195  
cu\_bad  
vtss\_phy\_statistic\_t, 187  
cu\_good  
vtss\_phy\_statistic\_t, 186  
cur\_version  
vtss\_restart\_status\_t, 258  
data  
vtss\_ace\_frame\_etype\_t, 31  
vtss\_ace\_frame\_ipv4\_t, 34  
vtss\_ace\_frame\_ipv6\_t, 38  
vtss\_mce\_key\_t, 119  
vtss\_qce\_frame\_etype\_t, 234  
vtss\_qce\_frame\_llc\_t, 238  
vtss\_qce\_frame\_snap\_t, 239  
vtss\_vce\_frame\_etype\_t, 294  
vtss\_vce\_frame\_llc\_t, 298  
vtss\_vce\_frame\_snap\_t, 298  
default\_dei  
vtss\_qos\_port\_conf\_t, 253  
default\_dpl  
vtss\_qos\_port\_conf\_t, 253  
default\_prio  
vtss\_qos\_port\_conf\_t, 252  
dei  
vtss\_ece\_outer\_tag\_t, 77  
vtss\_ece\_tag\_t, 80  
vtss\_evc\_inner\_tag\_t, 88  
vtss\_qce\_action\_t, 233  
vtss\_qce\_tag\_t, 245  
vtss\_vce\_tag\_t, 304  
vtss\_vlan\_tag\_t, 310  
dei\_colouring  
vtss\_evc\_port\_conf\_t, 90  
destination  
vtss\_ipv4\_uc\_t, 102  
vtss\_ipv6\_uc\_t, 104  
vtss\_irq\_conf\_t, 105  
vtss\_mac\_table\_entry\_t, 114  
dgroup\_no  
vtss\_dgroup\_port\_conf\_t, 65  
dip  
vtss\_ace\_frame\_arp\_t, 30  
vtss\_ace\_frame\_ipv4\_t, 33  
dir  
vtss\_ece\_action\_t, 68  
disable  
vtss\_phy\_mac\_serdes\_pcs\_ctrl\_t, 177  
discard  
vtss\_learn\_mode\_t, 112  
divider  
vtss\_sync\_clock\_out\_t, 269  
dma\_enable  
vtss\_packet\_dma\_conf\_t, 124  
dmac  
vtss\_ace\_frame\_etype\_t, 30  
vtss\_ace\_frame\_llc\_t, 41  
vtss\_ace\_frame\_snap\_t, 42  
dmac\_bc  
vtss\_ace\_t, 47  
vtss\_ece\_mac\_t, 76  
vtss\_qce\_mac\_t, 242  
vtss\_vce\_mac\_t, 302  
dmac\_dip  
vtss\_evc\_port\_conf\_t, 91  
vtss\_qos\_port\_conf\_t, 256  
vtss\_vcl\_port\_conf\_t, 305  
dmac\_enable  
vtss\_aggr\_mode\_t, 56  
dmac\_match  
vtss\_ace\_frame\_arp\_t, 29  
dmac\_mc  
vtss\_ace\_t, 47  
vtss\_ece\_mac\_t, 76

vtss\_qce\_mac\_t, 242  
vtss\_vce\_mac\_t, 302  
dot1dTpPortInDiscards  
    vtss\_port\_bridge\_counters\_t, 198  
dot3InPauseFrames  
    vtss\_port\_ethernet\_like\_counters\_t, 207  
dot3OutPauseFrames  
    vtss\_port\_ethernet\_like\_counters\_t, 208  
dp  
    vtss\_packet\_tx\_info\_t, 160  
    vtss\_qce\_action\_t, 232  
dp\_enable  
    vtss\_qce\_action\_t, 232  
dp\_level\_map  
    vtss\_qos\_port\_conf\_t, 254  
dport  
    vtss\_ace\_frame\_ipv4\_t, 34  
    vtss\_ace\_frame\_ipv6\_t, 38  
    vtss\_ece\_frame\_ipv4\_t, 71  
    vtss\_ece\_frame\_ipv6\_t, 73  
    vtss\_qce\_frame\_ipv4\_t, 236  
    vtss\_qce\_frame\_ipv6\_t, 237  
    vtss\_vce\_frame\_ipv4\_t, 295  
    vtss\_vce\_frame\_ipv6\_t, 297  
ds  
    vtss\_ace\_frame\_ipv4\_t, 33  
    vtss\_ace\_frame\_ipv6\_t, 38  
dscp  
    vtss\_ece\_frame\_ipv4\_t, 70  
    vtss\_ece\_frame\_ipv6\_t, 72  
    vtss\_qce\_action\_t, 232  
    vtss\_qce\_frame\_ipv4\_t, 235  
    vtss\_qce\_frame\_ipv6\_t, 237  
    vtss\_vce\_frame\_ipv4\_t, 295  
    vtss\_vce\_frame\_ipv6\_t, 296  
dscp\_class\_enable  
    vtss\_qos\_port\_conf\_t, 254  
dscp\_dp\_level\_map  
    vtss\_qos\_conf\_t, 247  
dscp\_emode  
    vtss\_qos\_port\_conf\_t, 254  
dscp\_enable  
    vtss\_qce\_action\_t, 232  
dscp\_mode  
    vtss\_qos\_port\_conf\_t, 254  
dscp\_qos\_class\_map  
    vtss\_qos\_conf\_t, 247  
dscp\_qos\_map  
    vtss\_qos\_conf\_t, 247  
dscp\_qos\_map\_dp1  
    vtss\_qos\_conf\_t, 247  
dscp\_remap  
    vtss\_qos\_conf\_t, 248  
dscp\_remap\_dp1  
    vtss\_qos\_conf\_t, 248  
dscp\_remark  
    vtss\_qos\_conf\_t, 248  
dscp\_translate  
    vtss\_qos\_port\_conf\_t, 254  
dst\_port\_mask  
    vtss\_packet\_tx\_info\_t, 154  
dual\_media\_fiber\_speed  
    port\_custom\_conf\_t, 25  
dwrr\_enable  
    vtss\_qos\_port\_conf\_t, 256  
ebs  
    vtss\_dlb\_policer\_conf\_t, 67  
eee\_ena  
    vtss\_eee\_port\_conf\_t, 81  
eee\_ena\_phy  
    vtss\_phy\_eee\_conf\_t, 170  
eee\_fast\_queues  
    vtss\_eee\_port\_conf\_t, 81  
eee\_mode  
    vtss\_phy\_eee\_conf\_t, 170  
eir  
    vtss\_dlb\_policer\_conf\_t, 67  
enable  
    port\_custom\_conf\_t, 23  
    vtss\_ace\_ptp\_t, 44  
    vtss\_ace\_sip\_smac\_t, 45  
    vtss\_dlb\_policer\_conf\_t, 66  
    vtss\_ece\_outer\_tag\_t, 77  
    vtss\_npi\_conf\_t, 123  
    vtss\_packet\_rx\_queue\_npi\_conf\_t, 150  
    vtss\_port\_clause\_37\_control\_t, 200  
    vtss\_sync\_clock\_in\_t, 268  
    vtss\_sync\_clock\_out\_t, 269  
    vtss\_ts\_ext\_clock\_mode\_t, 274  
enabled  
    vtss\_sgpi\_port\_conf\_t, 265  
ethernet  
    vtss\_ace\_frame\_arp\_t, 29  
ethernet\_like  
    vtss\_port\_counters\_t, 206  
etype  
    vtss\_ace\_frame\_etype\_t, 31  
    vtss\_ace\_t, 48  
    vtss\_packet\_rx\_meta\_t, 143  
    vtss\_qce\_frame\_etype\_t, 234  
    vtss\_qce\_key\_t, 240  
    vtss\_vce\_frame\_etype\_t, 293  
    vtss\_vce\_key\_t, 300  
evc  
    vtss\_port\_counters\_t, 207  
evc\_id  
    vtss\_ece\_action\_t, 69  
evc\_police  
    vtss\_acl\_action\_t, 52  
evc\_policer\_id  
    vtss\_acl\_action\_t, 52  
exc\_col\_cont

port\_custom\_conf\_t, 25  
 vtss\_port\_conf\_t, 204  
 excess\_enable  
 vtss\_qos\_port\_conf\_t, 252  
 external  
 vtss\_irq\_conf\_t, 105  
 FALSE  
 types.h, 356  
 fan\_low\_pol  
 vtss\_fan\_conf\_t, 92  
 fan\_open\_col  
 vtss\_fan\_conf\_t, 92  
 fan\_pwm\_freq  
 vtss\_fan\_conf\_t, 92  
 far\_end\_enable  
 vtss\_phy\_loopback\_t, 174  
 fast\_link\_stat\_ena  
 vtss\_phy\_mac\_serdes\_pcs\_ctrl\_t, 179  
 fcs  
 vtss\_packet\_rx\_meta\_t, 143  
 fdx  
 port\_custom\_conf\_t, 24  
 vtss\_phy\_forced\_t, 172  
 vtss\_port\_clause\_37\_adv\_t, 199  
 vtss\_port\_conf\_t, 203  
 vtss\_port\_sgmii\_aneg\_t, 227  
 vtss\_port\_status\_t, 229  
 fdx\_gap  
 vtss\_port\_frame\_gaps\_t, 212  
 fiber  
 vtss\_port\_status\_t, 230  
 file  
 vtss\_api\_lock\_t, 59  
 fill\_level  
 vtss\_eee\_port\_counter\_t, 83  
 fill\_level\_get  
 vtss\_eee\_port\_counter\_t, 82  
 fill\_level\_thres  
 vtss\_eee\_port\_counter\_t, 83  
 filter  
 vtss\_packet\_port\_filter\_t, 127  
 flf  
 vtss\_phy\_conf\_t, 168  
 flow\_control  
 port\_custom\_conf\_t, 24  
 vtss\_policer\_ext\_t, 196  
 vtss\_port\_conf\_t, 203  
 force  
 vtss\_phy\_reset\_conf\_t, 184  
 force\_adv\_ability  
 vtss\_phy\_mac\_serdes\_pcs\_ctrl\_t, 177  
 vtss\_phy\_media\_serdes\_pcs\_ctrl\_t, 180  
 force\_ams\_sel  
 vtss\_phy\_conf\_t, 169  
 force\_fefi  
 vtss\_phy\_media\_serdes\_pcs\_ctrl\_t, 181  
 force\_fefi\_value  
 vtss\_phy\_media\_serdes\_pcs\_ctrl\_t, 181

force\_hls  
 vtss\_phy\_media\_serdes\_pcs\_ctrl\_t, 181  
 forced  
 vtss\_phy\_conf\_t, 167  
 fragment  
 vtss\_ace\_frame\_ipv4\_t, 32  
 vtss\_ece\_frame\_ipv4\_t, 70  
 vtss\_qce\_frame\_ipv4\_t, 235  
 vtss\_vce\_frame\_ipv4\_t, 294  
 frame  
 vtss\_ace\_t, 49  
 vtss\_ece\_key\_t, 75  
 vtss\_qce\_key\_t, 241  
 vtss\_vce\_key\_t, 301  
 frame\_gaps  
 vtss\_port\_conf\_t, 202  
 frame\_length\_chk  
 port\_custom\_conf\_t, 26  
 vtss\_port\_conf\_t, 204  
 frame\_rate  
 vtss\_policer\_ext\_t, 196  
 frame\_type  
 vtss\_vlan\_port\_conf\_t, 309  
 frames  
 vtss\_counter\_pair\_t, 62  
 freq  
 vtss\_phy\_clock\_conf\_t, 165  
 vtss\_ts\_ext\_clock\_mode\_t, 275  
 frm\_len  
 vtss\_packet\_tx\_info\_t, 155  
 fsm\_lock  
 vtss\_lcpll\_status\_t, 110  
 fsm\_stat  
 vtss\_lcpll\_status\_t, 111  
 full  
 vtss\_debug\_info\_t, 63  
 function  
 vtss\_api\_lock\_t, 59  
 fwd\_enable  
 vtss\_mirror\_conf\_t, 121  
 gain\_stat  
 vtss\_lcpll\_status\_t, 111  
 garp\_cpu\_only  
 vtss\_packet\_rx\_reg\_t, 151  
 garp\_queue  
 vtss\_packet\_rx\_queue\_map\_t, 148  
 garp\_reg  
 vtss\_packet\_rx\_port\_conf\_t, 146  
 generate  
 vtss\_port\_flow\_control\_conf\_t, 211  
 generate\_pause  
 vtss\_aneg\_t, 58  
 glag\_no  
 vtss\_packet\_rx\_info\_t, 134  
 group  
 vtss\_debug\_info\_t, 63  
 group\_id  
 vtss\_vlan\_trans\_grp2vlan\_conf\_t, 311

vtss\_vlan\_trans\_port2grp\_conf\_t, 312  
grp\_map  
    vtss\_packet\_rx\_conf\_t, 130

hdx  
    vtss\_port\_clause\_37\_adv\_t, 199  
    vtss\_port\_sgmii\_aneg\_t, 227

hdx\_gap\_1  
    vtss\_port\_frame\_gaps\_t, 212

hdx\_gap\_2  
    vtss\_port\_frame\_gaps\_t, 212

header  
    vtss\_ace\_ptp\_t, 44

high  
    vtss\_vcap\_udp\_tcp\_t, 289  
    vtss\_vcap\_vr\_t, 291

hints  
    vtss\_packet\_rx\_info\_t, 133

hw\_cnt  
    vtss\_os\_timestamp\_t, 124

hw\_tstamp  
    vtss\_packet\_rx\_info\_t, 138

hw\_tstamp\_decoded  
    vtss\_packet\_rx\_info\_t, 139

i16  
    types.h, 373

i32  
    types.h, 373

i64  
    types.h, 373

i8  
    types.h, 373

i\_cpu\_en  
    vtss\_phy\_reset\_conf\_t, 185

ib\_cterm\_ena  
    vtss\_serdes\_macro\_conf\_t, 262

id  
    vtss\_ace\_t, 46  
    vtss\_ece\_t, 78  
    vtss\_mce\_t, 120  
    vtss\_qce\_t, 243  
    vtss\_ts\_timestamp\_t, 279  
    vtss\_vce\_t, 303

if\_group  
    vtss\_port\_counters\_t, 206

if\_type  
    vtss\_port\_conf\_t, 202

ifInBroadcastPkts  
    vtss\_port\_if\_group\_counters\_t, 214

ifInDiscards  
    vtss\_port\_if\_group\_counters\_t, 214

ifInErrors  
    vtss\_port\_if\_group\_counters\_t, 214

ifInMulticastPkts  
    vtss\_port\_if\_group\_counters\_t, 213

ifInNUcastPkts  
    vtss\_port\_if\_group\_counters\_t, 214

ifInOctets  
    vtss\_port\_if\_group\_counters\_t, 214

ifInUcastPkts  
    vtss\_port\_if\_group\_counters\_t, 213

ifOutBroadcastPkts  
    vtss\_port\_if\_group\_counters\_t, 215

ifOutDiscards  
    vtss\_port\_if\_group\_counters\_t, 215

ifOutErrors  
    vtss\_port\_if\_group\_counters\_t, 216

ifOutMulticastPkts  
    vtss\_port\_if\_group\_counters\_t, 215

ifOutNUcastPkts  
    vtss\_port\_if\_group\_counters\_t, 215

ifOutOctets  
    vtss\_port\_if\_group\_counters\_t, 214

ifOutUcastPkts  
    vtss\_port\_if\_group\_counters\_t, 215

ifih  
    vtss\_packet\_tx\_ifh\_t, 152

igmp\_cpu\_only  
    vtss\_packet\_rx\_reg\_t, 151

igmp\_queue  
    vtss\_packet\_rx\_queue\_map\_t, 148

in\_range  
    vtss\_vcap\_udp\_tcp\_t, 288

ingress\_filter  
    vtss\_vlan\_port\_conf\_t, 309

inhibit\_odd\_start  
    vtss\_phy\_mac\_serdes\_pcs\_cntl\_t, 179  
    vtss\_phy\_media\_serdes\_pcs\_cntl\_t, 181

inner\_tag  
    vtss\_evc\_pb\_conf\_t, 90  
    vtss\_evc\_port\_conf\_t, 91

inst  
    vtss\_api\_lock\_t, 59

int\_fmt  
    vtss\_ts\_internal\_mode\_t, 276

int\_pol\_high  
    vtss\_sgpi\_port\_conf\_t, 265

ip  
    vtss\_ace\_frame\_arp\_t, 29

ipmc\_ctrl\_cpu\_copy  
    vtss\_packet\_rx\_reg\_t, 151

ipmc\_ctrl\_queue  
    vtss\_packet\_rx\_queue\_map\_t, 148

ipv4  
    vtss\_ace\_t, 48  
    vtss\_ece\_key\_t, 74  
    vtss\_ip\_addr\_t, 99  
    vtss\_qce\_key\_t, 241  
    vtss\_vce\_key\_t, 300

ipv4\_uc  
    vtss\_routing\_entry\_t, 259

ipv4uc\_received\_frames  
    vtss\_l3\_counters\_t, 108

ipv4uc\_received\_octets  
    vtss\_l3\_counters\_t, 108

ipv4uc\_transmitted\_frames

vtss\_l3\_counters\_t, 109  
 ipv4uc\_transmitted\_octets  
     vtss\_l3\_counters\_t, 108  
 ipv6  
     vtss\_ace\_t, 49  
     vtss\_ece\_key\_t, 75  
     vtss\_ip\_addr\_t, 99  
     vtss\_qce\_key\_t, 241  
     vtss\_vce\_key\_t, 301  
 ipv6\_uc  
     vtss\_routing\_entry\_t, 260  
 ipv6uc\_received\_frames  
     vtss\_l3\_counters\_t, 108  
 ipv6uc\_received\_octets  
     vtss\_l3\_counters\_t, 108  
 ipv6uc\_transmitted\_frames  
     vtss\_l3\_counters\_t, 109  
 ipv6uc\_transmitted\_octets  
     vtss\_l3\_counters\_t, 109  
 isdx  
     vtss\_packet\_rx\_info\_t, 140  
     vtss\_packet\_tx\_info\_t, 159  
 isdx\_dont\_use  
     vtss\_packet\_tx\_info\_t, 160  
 ivid  
     vtss\_evc\_pb\_conf\_t, 89  
 key  
     vtss\_ece\_t, 78  
     vtss\_mce\_t, 120  
     vtss\_qce\_t, 244  
     vtss\_vce\_t, 303  
 l2\_types.h  
     vtss\_sflow\_type\_t, 315  
 latch\_timestamp  
     vtss\_packet\_tx\_info\_t, 158  
 layer  
     vtss\_debug\_info\_t, 63  
 lb\_type  
     vtss\_phy\_api.h, 578  
 learn  
     vtss\_acl\_action\_t, 52  
     vtss\_packet\_rx\_header\_t, 131  
 learn\_queue  
     vtss\_packet\_rx\_queue\_map\_t, 148  
 learned  
     vtss\_mac\_table\_status\_t, 115  
 learning  
     vtss\_evc\_conf\_t, 86  
     vtss\_vlan\_vid\_conf\_t, 313  
 length  
     vtss\_ace\_frame\_arp\_t, 29  
     vtss\_packet\_rx\_header\_t, 130  
     vtss\_packet\_rx\_info\_t, 133  
     vtss\_packet\_rx\_meta\_t, 144  
     vtss\_packet\_tx\_ifh\_t, 152  
     vtss\_phy\_veriphy\_result\_t, 192  
 level  
     vtss\_phy\_power\_status\_t, 183  
     vtss\_policer\_t, 197  
     vtss\_shaper\_t, 267  
     vtss\_trace\_conf\_t, 274  
 line  
     vtss\_api\_lock\_t, 59  
 line\_rate  
     vtss\_dlb\_policer\_conf\_t, 67  
 link  
     vtss\_phy\_veriphy\_result\_t, 192  
     vtss\_port\_sgmii\_aneg\_t, 227  
     vtss\_port\_status\_t, 229  
 link\_change  
     vtss\_intr\_t, 98  
 link\_down  
     vtss\_port\_status\_t, 229  
 llabs  
     vtss\_os\_ecos.h, 517  
 llc  
     vtss\_ace\_frame\_llc\_t, 42  
     vtss\_ace\_t, 48  
     vtss\_qce\_key\_t, 241  
     vtss\_vce\_key\_t, 300  
 lock\_status  
     vtss\_lcpll\_status\_t, 110  
 locked  
     vtss\_mac\_table\_entry\_t, 114  
 loop  
     vtss\_port\_conf\_t, 205  
 low  
     vtss\_vcap\_udp\_tcp\_t, 288  
     vtss\_vcap\_vr\_t, 291  
 lp\_advertisement  
     vtss\_eee\_port\_conf\_t, 81  
 lrn\_all\_queue  
     vtss\_packet\_rx\_queue\_map\_t, 149  
 MAC\_ADDR\_BROADCAST  
     types.h, 364  
 MAX\_CFG\_BUF\_SIZE  
     vtss\_phy\_api.h, 558  
 MAX\_STAT\_BUF\_SIZE  
     vtss\_phy\_api.h, 558  
 MAX\_WOL\_MAC\_ADDR\_SIZE  
     vtss\_phy\_api.h, 569  
 MAX\_WOL\_PASSWD\_SIZE  
     vtss\_phy\_api.h, 569  
 MIN\_WOL\_PASSWD\_SIZE  
     vtss\_phy\_api.h, 569  
 mac  
     vtss\_ece\_key\_t, 74  
     vtss\_qce\_key\_t, 240  
     vtss\_vce\_key\_t, 299  
     vtss\_vid\_mac\_t, 306  
 mac\_if  
     vtss\_phy\_reset\_conf\_t, 184  
 mac\_if\_pcs  
     vtss\_phy\_conf\_t, 169  
 mac\_serdes\_equipment\_enable

vtss\_phy\_loopback\_t, 175  
mac\_serdes\_facility\_enable  
    vtss\_phy\_loopback\_t, 175  
mac\_serdes\_input\_enable  
    vtss\_phy\_loopback\_t, 175  
mac\_vid\_queue  
    vtss\_packet\_rx\_queue\_map\_t, 148  
magic\_pkt\_cnt  
    vtss\_phy\_wol\_conf\_t, 194  
map  
    vtss\_packet\_rx\_conf\_t, 129  
mask  
    vtss\_vcap\_ip\_t, 280  
    vtss\_vcap\_u128\_t, 281  
    vtss\_vcap\_u16\_t, 282  
    vtss\_vcap\_u24\_t, 283  
    vtss\_vcap\_u32\_t, 284  
    vtss\_vcap\_u40\_t, 285  
    vtss\_vcap\_u48\_t, 286  
    vtss\_vcap\_u8\_t, 287  
    vtss\_vcap\_vid\_t, 290  
    vtss\_vcap\_vr\_t, 291  
masquerade\_port  
    vtss\_packet\_tx\_info\_t, 161  
master  
    vtss\_phy\_conf\_1g\_t, 166  
    vtss\_phy\_status\_1g\_t, 189  
master\_cfg\_fault  
    vtss\_phy\_status\_1g\_t, 188  
max\_frame\_length  
    vtss\_port\_conf\_t, 203  
max\_length  
    port\_custom\_conf\_t, 25  
max\_tags  
    port\_custom\_conf\_t, 25  
    vtss\_port\_conf\_t, 204  
mdi  
    vtss\_phy\_conf\_t, 168  
mdi\_cross  
    vtss\_port\_status\_t, 230  
media\_if  
    vtss\_phy\_reset\_conf\_t, 184  
media\_if\_pcs  
    vtss\_phy\_conf\_t, 169  
media\_mac\_serdes\_crc  
    vtss\_phy\_statistic\_t, 188  
media\_mac\_serdes\_good  
    vtss\_phy\_statistic\_t, 187  
media\_serdes\_equipment\_enable  
    vtss\_phy\_loopback\_t, 176  
media\_serdes\_facility\_enable  
    vtss\_phy\_loopback\_t, 176  
media\_serdes\_input\_enable  
    vtss\_phy\_loopback\_t, 175  
miim\_addr  
    vtss\_port\_map\_t, 217  
miim\_chip\_no  
    vtss\_port\_map\_t, 217  
                miim\_controller  
                vtss\_port\_map\_t, 217  
                miim\_read  
                vtss\_init\_conf\_t, 94  
                miim\_write  
                vtss\_init\_conf\_t, 94  
                mirror  
                vtss\_acl\_action\_t, 53  
                vtss\_vlan\_vid\_conf\_t, 313  
                mld\_cpu\_only  
                vtss\_packet\_rx\_reg\_t, 151  
                mmd\_read  
                vtss\_init\_conf\_t, 94  
                mmd\_read\_inc  
                vtss\_init\_conf\_t, 94  
                mmd\_write  
                vtss\_init\_conf\_t, 95  
                mode  
                vtss\_phy\_conf\_t, 167  
                vtss\_phy\_led\_mode\_select\_t, 173  
                vtss\_phy\_power\_conf\_t, 182  
                vtss\_sgpio\_port\_conf\_t, 265  
                vtss\_ts\_operation\_mode\_t, 277  
                moved  
                vtss\_mac\_table\_status\_t, 116  
                mux\_mode  
                vtss\_init\_conf\_t, 96  
                nanoseconds  
                vtss\_timestamp\_t, 273  
                near\_end\_enable  
                vtss\_phy\_loopback\_t, 174  
                network  
                vtss\_evc\_conf\_t, 86  
                vtss\_ipv4\_uc\_t, 102  
                vtss\_ipv6\_uc\_t, 104  
                next\_page  
                vtss\_port\_clause\_37\_adv\_t, 200  
                nni  
                vtss\_evc\_pb\_conf\_t, 89  
                no\_wait  
                vtss\_packet\_rx\_meta\_t, 142  
                now  
                vtss\_mtimer\_t, 122  
                npi  
                vtss\_packet\_rx\_queue\_conf\_t, 147  
                number  
                vtss\_phy\_led\_mode\_select\_t, 173  
                oam\_info  
                vtss\_packet\_rx\_info\_t, 140  
                oam\_info\_decoded  
                vtss\_packet\_rx\_info\_t, 140  
                oam\_type  
                vtss\_packet\_tx\_info\_t, 159  
                ob\_post0  
                serdes\_fields\_t, 27  
                ob\_sr  
                serdes\_fields\_t, 27

obey  
     vtss\_port\_flow\_control\_conf\_t, 210  
 obey\_pause  
     vtss\_aneg\_t, 58  
 one\_pps\_mode  
     vtss\_ts\_ext\_clock\_mode\_t, 274  
 oper\_up  
     port\_custom\_conf\_t, 26  
 optimized\_for\_power  
     vtss\_eee\_port\_conf\_t, 82  
 options  
     vtss\_ace\_frame\_ipv4\_t, 33  
     vtss\_vce\_frame\_ipv4\_t, 295  
 options.h  
     VTSS\_ARCH\_CARACAL, 317  
     VTSS\_ARCH\_LUTON26, 318  
     VTSS\_CHIP CU PHY, 331  
     VTSS\_FEATURE\_ACL\_V2, 328  
     VTSS\_FEATURE\_ACL, 328  
     VTSS\_FEATURE\_CLAUSE\_37, 319  
     VTSS\_FEATURE\_EEE, 327  
     VTSS\_FEATURE\_EVC, 318  
     VTSS\_FEATURE\_EXC\_COL\_CONT, 319  
     VTSS\_FEATURE\_FAN, 327  
     VTSS\_FEATURE\_INTERRUPTS, 330  
     VTSS\_FEATURE\_IPV4\_MC\_SIP, 326  
     VTSS\_FEATURE\_IPV6\_MC\_SIP, 326  
     VTSS\_FEATURE\_IRQ\_CONTROL, 329  
     VTSS\_FEATURE\_LAYER2, 325  
     VTSS\_FEATURE\_LED\_POW\_REDUC, 329  
     VTSS\_FEATURE\_MAC\_AGE\_AUTO, 326  
     VTSS\_FEATURE\_MAC\_CPU\_QUEUE, 327  
     VTSS\_FEATURE\_MIRROR\_CPU, 330  
     VTSS\_FEATURE\_MISC, 318  
     VTSS\_FEATURE\_NPI, 329  
     VTSS\_FEATURE\_PACKET\_GROUPING, 325  
     VTSS\_FEATURE\_PACKET\_PORT\_REG, 325  
     VTSS\_FEATURE\_PACKET\_RX, 325  
     VTSS\_FEATURE\_PACKET\_TX, 324  
     VTSS\_FEATURE\_PACKET, 324  
     VTSS\_FEATURE\_PORT\_CNT\_BRIDGE, 319  
     VTSS\_FEATURE\_PORT\_CONTROL, 319  
     VTSS\_FEATURE\_PORT\_MUX, 327  
     VTSS\_FEATURE\_PVLAN, 325  
     VTSS\_FEATURE\_QCL\_DMAC\_DIP, 320  
     VTSS\_FEATURE\_QCL\_KEY\_S\_TAG, 320  
     VTSS\_FEATURE\_QCL\_PCP\_DEI\_ACTION, 320  
     VTSS\_FEATURE\_QCL\_POLICY\_ACTION, 321  
     VTSS\_FEATURE\_QCL\_V2, 320  
     VTSS\_FEATURE\_QCL, 320  
     VTSS\_FEATURE\_QOS\_CLASSIFICATION\_V2,  
         323  
     VTSS\_FEATURE\_QOS\_DSCP\_CLASS\_DP\_A←  
         WARE, 323  
     VTSS\_FEATURE\_QOS\_DSCP\_REMARK\_DP←  
         WARE, 324  
     VTSS\_FEATURE\_QOS\_DSCP\_REMARK\_V2,  
         324  
     VTSS\_FEATURE\_QOS\_DSCP\_REMARK, 324  
     VTSS\_FEATURE\_QOS\_EGRESS\_QUEUE\_SH←  
         APERS\_EB, 323  
     VTSS\_FEATURE\_QOS\_EGRESS\_QUEUE\_SH←  
         APERS, 323  
     VTSS\_FEATURE\_QOS\_POLICER\_BC\_SWITCH,  
         321  
     VTSS\_FEATURE\_QOS\_POLICER\_DLDB, 318  
     VTSS\_FEATURE\_QOS\_POLICER\_MC\_SWITCH,  
         321  
     VTSS\_FEATURE\_QOS\_POLICER\_UC\_SWITCH,  
         321  
     VTSS\_FEATURE\_QOS\_PORT\_POLICER\_EXT←  
         FPS, 322  
     VTSS\_FEATURE\_QOS\_PORT\_POLICER\_EXT←  
         FC, 322  
     VTSS\_FEATURE\_QOS\_PORT\_POLICER\_EXT,  
         321  
     VTSS\_FEATURE\_QOS\_QUEUE\_POLICER, 322  
     VTSS\_FEATURE\_QOS\_QUEUE\_TX, 322  
     VTSS\_FEATURE\_QOS\_SCHEDULER\_V2, 322  
     VTSS\_FEATURE\_QOS\_TAG\_REMARK\_V2, 323  
     VTSS\_FEATURE\_QOS, 319  
     VTSS\_FEATURE\_SERDES\_MACRO\_SETTIN←  
         GS, 330  
     VTSS\_FEATURE\_SERIAL\_GPIO, 318  
     VTSS\_FEATURE\_SFLOW, 330  
     VTSS\_FEATURE\_SYNC, 329  
     VTSS\_FEATURE\_TIMESTAMP\_LATENCY\_CO←  
         MP, 329  
     VTSS\_FEATURE\_TIMESTAMP\_ONE\_STEP, 328  
     VTSS\_FEATURE\_TIMESTAMP, 328  
     VTSS\_FEATURE\_VCAP, 327, 332  
     VTSS\_FEATURE\_VCL, 328  
     VTSS\_FEATURE\_VLAN\_PORT\_V2, 326  
     VTSS\_FEATURE\_VLAN\_TRANSLATION, 330  
     VTSS\_FEATURE\_VLAN\_TX\_TAG, 326  
     VTSS\_FEATURE\_WARM\_START, 331  
     VTSS\_OPT\_TRACE, 331  
     VTSS\_OPT\_VAUI\_EQ\_CTRL, 331  
     VTSS\_PHY\_OPT\_VERIPHY, 331  
 out\_of\_range  
     vtss\_rcpll\_status\_t, 257  
 outer\_tag  
     vtss\_ece\_action\_t, 69  
 PORT\_CAP\_100M\_FDX  
     port.h, 337  
 PORT\_CAP\_100M\_HDX  
     port.h, 337  
 PORT\_CAP\_10G\_FDX  
     port.h, 338  
 PORT\_CAP\_10M\_FDX  
     port.h, 337  
 PORT\_CAP\_10M\_HDX  
     port.h, 337  
 PORT\_CAP\_1G\_FDX  
     port.h, 337  
 PORT\_CAP\_1G\_PHY

port.h, 342  
PORT\_CAP\_2\_5G\_FDX  
    port.h, 338  
PORT\_CAP\_2\_5G\_TRI\_SPEED\_COPPER  
    port.h, 345  
PORT\_CAP\_2\_5G\_TRI\_SPEED\_FDX  
    port.h, 345  
PORT\_CAP\_2\_5G\_TRI\_SPEED  
    port.h, 345  
PORT\_CAP\_5G\_FDX  
    port.h, 338  
PORT\_CAP\_ANY\_FIBER  
    port.h, 343  
PORT\_CAP\_AUTONEG  
    port.h, 336  
PORT\_CAP\_COPPER  
    port.h, 338  
PORT\_CAP\_DUAL\_COPPER\_100FX  
    port.h, 341  
PORT\_CAP\_DUAL\_COPPER  
    port.h, 339  
PORT\_CAP\_DUAL\_FIBER\_1000X  
    port.h, 344  
PORT\_CAP\_DUAL\_FIBER\_100FX  
    port.h, 340  
PORT\_CAP\_DUAL\_FIBER  
    port.h, 339  
PORT\_CAP\_DUAL\_SFP\_DETECT  
    port.h, 341  
PORT\_CAP\_FIBER  
    port.h, 339  
PORT\_CAP\_FLOW\_CTRL  
    port.h, 338  
PORT\_CAP\_HDX  
    port.h, 341  
PORT\_CAP\_NONE  
    port.h, 336  
PORT\_CAP\_SD\_ENABLE  
    port.h, 339  
PORT\_CAP\_SD\_HIGH  
    port.h, 339  
PORT\_CAP\_SD\_INTERNAL  
    port.h, 340  
PORT\_CAP\_SFP\_1G  
    port.h, 344  
PORT\_CAP\_SFP\_2\_5G  
    port.h, 345  
PORT\_CAP\_SFP\_DETECT  
    port.h, 340  
PORT\_CAP\_SFP\_ONLY  
    port.h, 341  
PORT\_CAP\_SFP\_SD\_HIGH  
    port.h, 345  
PORT\_CAP\_SPEED\_DUAL\_ANY\_FIBER\_FIXED\_SPEED  
    port.h, 343  
PORT\_CAP\_SPEED\_DUAL\_ANY\_FIBER  
    port.h, 344  
PORT\_CAP\_STACKING  
    port.h, 341  
PORT\_CAP\_TRI\_SPEED\_COPPER  
    port.h, 342  
PORT\_CAP\_TRI\_SPEED\_DUAL\_ANY\_FIBER\_FIXED\_SFP\_SPEED  
    port.h, 344  
PORT\_CAP\_TRI\_SPEED\_DUAL\_ANY\_FIBER  
    port.h, 344  
PORT\_CAP\_TRI\_SPEED\_DUAL\_COPPER  
    port.h, 343  
PORT\_CAP\_TRI\_SPEED\_DUAL\_FIBER\_100FX  
    port.h, 343  
PORT\_CAP\_TRI\_SPEED\_DUAL\_FIBER  
    port.h, 343  
PORT\_CAP\_TRI\_SPEED\_FDX  
    port.h, 342  
PORT\_CAP\_TRI\_SPEED\_FIBER  
    port.h, 342  
PORT\_CAP\_TRI\_SPEED  
    port.h, 342  
PORT\_CAP\_VTSS\_10G\_PHY  
    port.h, 340  
PORT\_CAP\_XAUI\_LANE\_FLIP  
    port.h, 340  
PRIi64  
    types.h, 354  
PRIu64  
    types.h, 354  
PRIx64  
    types.h, 354  
part\_number  
    vtss\_chip\_id\_t, 61  
    vtss\_phy\_type\_t, 190  
passwd  
    vtss\_secure\_on\_passwd\_t, 261  
pb  
    vtss\_evc\_conf\_t, 86  
pcp  
    vtss\_ece\_outer\_tag\_t, 77  
    vtss\_ece\_tag\_t, 79  
    vtss\_evc\_inner\_tag\_t, 88  
    vtss\_qce\_action\_t, 233  
    vtss\_qce\_tag\_t, 245  
    vtss\_vce\_tag\_t, 304  
    vtss\_vlan\_tag\_t, 310  
pcp\_dei\_enable  
    vtss\_qce\_action\_t, 232  
pcp\_dei\_preserve  
    vtss\_ece\_outer\_tag\_t, 77  
    vtss\_evc\_inner\_tag\_t, 88  
pd\_enable  
    vtss\_phy\_mac\_serdes\_pcs\_ctrl\_t, 177  
pdu\_offset  
    vtss\_packet\_tx\_info\_t, 161  
pfc  
    port\_custom\_conf\_t, 24  
    vtss\_port\_flow\_control\_conf\_t, 211

phy.h  
 VTSS\_PHY\_POWER\_ACTIPHY\_BIT, 333  
 VTSS\_PHY\_POWER\_DYNAMIC\_BIT, 333  
 vtss\_phy\_power\_mode\_t, 333  
 vtss\_phy\_veriphy\_status\_t, 334  
 phy\_api\_base\_no  
 vtss\_phy\_type\_t, 191  
 pi  
 vtss\_init\_conf\_t, 96  
 pkt\_mode  
 vtss\_phy\_reset\_conf\_t, 184  
 police  
 vtss\_acl\_action\_t, 52  
 policer\_bc  
 vtss\_qos\_conf\_t, 250  
 policer\_bc\_frame\_rate  
 vtss\_qos\_conf\_t, 250  
 policer\_bc\_mode  
 vtss\_qos\_conf\_t, 250  
 policer\_ext\_port  
 vtss\_qos\_port\_conf\_t, 251  
 policer\_mc  
 vtss\_qos\_conf\_t, 249  
 policer\_mc\_frame\_rate  
 vtss\_qos\_conf\_t, 249  
 policer\_mc\_mode  
 vtss\_qos\_conf\_t, 249  
 policer\_no  
 vtss\_acl\_action\_t, 52  
 policer\_port  
 vtss\_qos\_port\_conf\_t, 251  
 policer\_queue  
 vtss\_qos\_port\_conf\_t, 252  
 policer\_uc  
 vtss\_qos\_conf\_t, 248  
 policer\_uc\_frame\_rate  
 vtss\_qos\_conf\_t, 249  
 policer\_uc\_mode  
 vtss\_qos\_conf\_t, 249  
 policy  
 vtss\_ace\_t, 46  
 policy\_no  
 vtss\_acl\_port\_conf\_t, 55  
 vtss\_ece\_action\_t, 69  
 vtss\_mce\_action\_t, 117  
 vtss\_qce\_action\_t, 233  
 vtss\_vce\_action\_t, 293  
 policy\_no\_enable  
 vtss\_qce\_action\_t, 233  
 pop\_cnt  
 vtss\_mce\_action\_t, 117  
 pop\_tag  
 vtss\_ece\_action\_t, 68  
 port.h  
 PORT\_CAP\_100M\_FDX, 337  
 PORT\_CAP\_100M\_HDX, 337  
 PORT\_CAP\_10G\_FDX, 338  
 PORT\_CAP\_10M\_FDX, 337  
 PORT\_CAP\_10M\_HDX, 337  
 PORT\_CAP\_1G\_FDX, 337  
 PORT\_CAP\_1G\_PHY, 342  
 PORT\_CAP\_2\_5G\_FDX, 338  
 PORT\_CAP\_2\_5G\_TRI\_SPEED\_COPPER, 345  
 PORT\_CAP\_2\_5G\_TRI\_SPEED\_FDX, 345  
 PORT\_CAP\_2\_5G\_TRI\_SPEED, 345  
 PORT\_CAP\_5G\_FDX, 338  
 PORT\_CAP\_ANY\_FIBER, 343  
 PORT\_CAP\_AUTONEG, 336  
 PORT\_CAP\_COPPER, 338  
 PORT\_CAP\_DUAL\_COPPER\_100FX, 341  
 PORT\_CAP\_DUAL\_COPPER, 339  
 PORT\_CAP\_DUAL\_FIBER\_1000X, 344  
 PORT\_CAP\_DUAL\_FIBER\_100FX, 340  
 PORT\_CAP\_DUAL\_FIBER, 339  
 PORT\_CAP\_DUAL\_SFP\_DETECT, 341  
 PORT\_CAP\_FIBER, 339  
 PORT\_CAP\_FLOW\_CTRL, 338  
 PORT\_CAP\_HDX, 341  
 PORT\_CAP\_NONE, 336  
 PORT\_CAP\_SD\_ENABLE, 339  
 PORT\_CAP\_SD\_HIGH, 339  
 PORT\_CAP\_SD\_INTERNAL, 340  
 PORT\_CAP\_SFP\_1G, 344  
 PORT\_CAP\_SFP\_2\_5G, 345  
 PORT\_CAP\_SFP\_DETECT, 340  
 PORT\_CAP\_SFP\_ONLY, 341  
 PORT\_CAP\_SFP\_SD\_HIGH, 345  
 PORT\_CAP\_SPEED\_DUAL\_ANY\_FIBER\_FIXE←  
 D\_SPEED, 343  
 PORT\_CAP\_SPEED\_DUAL\_ANY\_FIBER, 344  
 PORT\_CAP\_STACKING, 341  
 PORT\_CAP\_TRI\_SPEED\_COPPER, 342  
 PORT\_CAP\_TRI\_SPEED\_DUAL\_ANY\_FIBER←  
 FIXED\_SFP\_SPEED, 344  
 PORT\_CAP\_TRI\_SPEED\_DUAL\_ANY\_FIBER,  
 344  
 PORT\_CAP\_TRI\_SPEED\_DUAL\_COPPER, 343  
 PORT\_CAP\_TRI\_SPEED\_DUAL\_FIBER\_100FX,  
 343  
 PORT\_CAP\_TRI\_SPEED\_DUAL\_FIBER, 343  
 PORT\_CAP\_TRI\_SPEED\_FDX, 342  
 PORT\_CAP\_TRI\_SPEED\_FIBER, 342  
 PORT\_CAP\_TRI\_SPEED, 342  
 PORT\_CAP\_VTSS\_10G\_PHY, 340  
 PORT\_CAP\_XAUI\_LANE\_FLIP, 340  
 port\_cap\_t, 346  
 vtss\_fiber\_port\_speed\_t, 346  
 vtss\_port\_speed\_t, 346  
 port\_action  
 vtss\_acl\_action\_t, 53  
 port\_cap\_t  
 port.h, 346  
 port\_cnt  
 vtss\_phy\_type\_t, 191  
 port\_conf  
 vtss\_sgpi\_conf\_t, 264

port\_custom\_conf\_t, 23  
adv\_dis, 25  
autoneg, 24  
dual\_media\_fiber\_speed, 25  
enable, 23  
exc\_col\_cont, 25  
fdx, 24  
flow\_control, 24  
frame\_length\_chk, 26  
max\_length, 25  
max\_tags, 25  
oper\_up, 26  
pfc, 24  
speed, 24  
port\_list  
    vtss\_ace\_t, 46  
    vtss\_acl\_action\_t, 53  
    vtss\_debug\_info\_t, 63  
    vtss\_ece\_key\_t, 74  
    vtss\_mce\_key\_t, 118  
    vtss\_qce\_key\_t, 240  
    vtss\_vce\_key\_t, 299  
port\_mask  
    vtss\_ts\_timestamp\_alloc\_t, 278  
port\_no  
    vtss\_eps\_port\_conf\_t, 85  
    vtss\_mirror\_conf\_t, 121  
    vtss\_npi\_conf\_t, 123  
    vtss\_packet\_frame\_info\_t, 125  
    vtss\_packet\_port\_info\_t, 128  
    vtss\_packet\_rx\_header\_t, 131  
    vtss\_packet\_rx\_info\_t, 133  
    vtss\_sync\_clock\_in\_t, 268  
port\_tx  
    vtss\_packet\_frame\_info\_t, 125  
port\_type  
    vtss\_vlan\_port\_conf\_t, 308  
ports  
    vtss\_vlan\_trans\_port2grp\_conf\_t, 312  
power\_down  
    vtss\_port\_conf\_t, 203  
ppr  
    vtss\_fan\_conf\_t, 92  
prefix\_size  
    vtss\_ip\_network\_t, 100  
    vtss\_ipv4\_network\_t, 101  
    vtss\_ipv6\_network\_t, 103  
prev\_version  
    vtss\_restart\_status\_t, 258  
prio  
    vtss\_ece\_action\_t, 69  
    vtss\_mce\_action\_t, 117  
    vtss\_qce\_action\_t, 231  
prio\_enable  
    vtss\_ece\_action\_t, 69  
    vtss\_mce\_action\_t, 117  
    vtss\_qce\_action\_t, 231  
prios  
    vtss\_qos\_conf\_t, 246  
prop  
    vtss\_port\_counters\_t, 206  
proto  
    vtss\_ace\_frame\_ipv4\_t, 33  
    vtss\_ace\_frame\_ipv6\_t, 37  
    vtss\_ece\_frame\_ipv4\_t, 71  
    vtss\_ece\_frame\_ipv6\_t, 72  
    vtss\_qce\_frame\_ipv4\_t, 235  
    vtss\_qce\_frame\_ipv6\_t, 237  
    vtss\_vce\_frame\_ipv4\_t, 295  
    vtss\_vce\_frame\_ipv6\_t, 296  
ptp  
    vtss\_ace\_frame\_etype\_t, 31  
    vtss\_ace\_frame\_ipv4\_t, 36  
    vtss\_ace\_frame\_ipv6\_t, 40  
ptp\_action  
    vtss\_acl\_action\_t, 53  
    vtss\_packet\_tx\_info\_t, 157  
ptp\_id  
    vtss\_packet\_tx\_info\_t, 157  
ptp\_timestamp  
    vtss\_packet\_tx\_info\_t, 158  
pvrid  
    vtss\_vlan\_port\_conf\_t, 308  
qos\_class\_map  
    vtss\_qos\_port\_conf\_t, 253  
qsgmii  
    vtss\_serdes\_macro\_conf\_t, 262  
queue  
    vtss\_packet\_rx\_conf\_t, 129  
queue\_mask  
    vtss\_packet\_rx\_header\_t, 131  
queue\_pct  
    vtss\_qos\_port\_conf\_t, 256  
r  
    vtss\_vcap\_vr\_t, 292  
rate  
    vtss\_acl\_policer\_conf\_t, 54  
    vtss\_policer\_t, 197  
    vtss\_shaper\_t, 267  
raw\_ident  
    vtss\_irq\_status\_t, 106  
raw\_mask  
    vtss\_irq\_status\_t, 107  
raw\_status  
    vtss\_irq\_status\_t, 107  
reg  
    vtss\_packet\_rx\_conf\_t, 129  
reg\_read  
    vtss\_init\_conf\_t, 93  
reg\_write  
    vtss\_init\_conf\_t, 94  
remote\_fault  
    vtss\_phy\_media\_serd\_pcs\_ctrl\_t, 180  
    vtss\_port\_clause\_37\_adv\_t, 199  
    vtss\_port\_status\_t, 229

replaced  
     vtss\_mac\_table\_status\_t, 115  
 req  
     vtss\_ace\_frame\_arp\_t, 28  
 restart  
     vtss\_phy\_mac\_serdes\_pcs\_cntl\_t, 177  
     vtss\_restart\_status\_t, 258  
 restart\_info\_port  
     vtss\_init\_conf\_t, 96  
 restart\_info\_src  
     vtss\_init\_conf\_t, 96  
 revision  
     vtss\_chip\_id\_t, 61  
     vtss\_phy\_type\_t, 190  
 rgmii  
     vtss\_phy\_reset\_conf\_t, 184  
 rgmii\_skew\_delay\_psec\_t  
     vtss\_phy\_api.h, 572  
 rmon  
     vtss\_port\_counters\_t, 206  
 route  
     vtss\_routing\_entry\_t, 260  
 rx\_clk\_skew\_ps  
     vtss\_phy\_rgmii\_conf\_t, 185  
 rx\_err\_cnt\_base\_tx  
     vtss\_phy\_statistic\_t, 187  
 rx\_etherStatsBroadcastPkts  
     vtss\_port\_rmon\_counters\_t, 220  
 rx\_etherStatsCRCAlignErrors  
     vtss\_port\_rmon\_counters\_t, 220  
 rx\_etherStatsDropEvents  
     vtss\_port\_rmon\_counters\_t, 219  
 rx\_etherStatsFragments  
     vtss\_port\_rmon\_counters\_t, 221  
 rx\_etherStatsJabbers  
     vtss\_port\_rmon\_counters\_t, 221  
 rx\_etherStatsMulticastPkts  
     vtss\_port\_rmon\_counters\_t, 220  
 rx\_etherStatsOctets  
     vtss\_port\_rmon\_counters\_t, 219  
 rx\_etherStatsOversizePkts  
     vtss\_port\_rmon\_counters\_t, 221  
 rx\_etherStatsPkts  
     vtss\_port\_rmon\_counters\_t, 220  
 rx\_etherStatsPkts1024to1518Octets  
     vtss\_port\_rmon\_counters\_t, 222  
 rx\_etherStatsPkts128to255Octets  
     vtss\_port\_rmon\_counters\_t, 222  
 rx\_etherStatsPkts1519toMaxOctets  
     vtss\_port\_rmon\_counters\_t, 222  
 rx\_etherStatsPkts256to511Octets  
     vtss\_port\_rmon\_counters\_t, 222  
 rx\_etherStatsPkts512to1023Octets  
     vtss\_port\_rmon\_counters\_t, 222  
 rx\_etherStatsPkts64Octets  
     vtss\_port\_rmon\_counters\_t, 221  
 rx\_etherStatsPkts65to127Octets  
     vtss\_port\_rmon\_counters\_t, 221  
 rx\_etherStatsUnderSizePkts  
     vtss\_port\_rmon\_counters\_t, 220  
 rx\_frames  
     vtss\_basic\_counters\_t, 60  
 rx\_green  
     vtss\_port\_evc\_counters\_t, 208  
 rx\_green\_discard  
     vtss\_port\_evc\_counters\_t, 209  
 rx\_prio  
     vtss\_port\_proprietary\_counters\_t, 218  
 rx\_red  
     vtss\_port\_evc\_counters\_t, 209  
 rx\_yellow  
     vtss\_port\_evc\_counters\_t, 209  
 rx\_yellow\_discard  
     vtss\_port\_evc\_counters\_t, 209  
 s\_etype  
     vtss\_vlan\_conf\_t, 307  
 s\_tag  
     vtss\_qce\_tag\_t, 245  
     vtss\_vce\_tag\_t, 305  
 s\_tagged  
     vtss\_ece\_tag\_t, 80  
 sampling\_rate  
     vtss\_sflow\_port\_conf\_t, 263  
 sd\_active\_high  
     vtss\_port\_conf\_t, 202  
 sd\_enable  
     vtss\_port\_conf\_t, 202  
 sd\_internal  
     vtss\_port\_conf\_t, 202  
 sec  
     vtss\_timeofday\_t, 271, 272  
 sec\_msb  
     vtss\_timestamp\_t, 272  
 seconds  
     vtss\_timestamp\_t, 273  
 secure\_on\_enable  
     vtss\_phy\_wol\_conf\_t, 193  
 select  
     vtss\_eee\_port\_state\_t, 84  
 seq\_zero  
     vtss\_ace\_frame\_ipv4\_t, 36  
     vtss\_ace\_frame\_ipv6\_t, 40  
 ser\_led\_frame\_rate  
     vtss\_phy\_enhanced\_led\_control\_t, 171  
 ser\_led\_output\_1  
     vtss\_phy\_enhanced\_led\_control\_t, 171  
 ser\_led\_output\_2  
     vtss\_phy\_enhanced\_led\_control\_t, 171  
 ser\_led\_select  
     vtss\_phy\_enhanced\_led\_control\_t, 171  
 serdes  
     vtss\_init\_conf\_t, 96  
     vtss\_port\_conf\_t, 205  
 serdes1g\_vdd  
     vtss\_serdes\_macro\_conf\_t, 261  
 serdes6g\_vdd

vtss\_serd़es\_macro\_conf\_t, 262  
serdes\_aneg\_ena  
    vtss\_phy\_mac\_serд\_pcs\_cntl\_t, 178  
serdes\_fields\_t, 26  
    ob\_post0, 27  
    ob\_sr, 27  
serdes\_pol\_inv\_in  
    vtss\_phy\_mac\_serд\_pcs\_cntl\_t, 178  
    vtss\_phy\_media\_serд\_pcs\_cntl\_t, 180  
serdes\_pol\_inv\_out  
    vtss\_phy\_mac\_serд\_pcs\_cntl\_t, 178  
    vtss\_phy\_media\_serд\_pcs\_cntl\_t, 180  
serdes\_tx\_bad  
    vtss\_phy\_statistic\_t, 187  
serdes\_tx\_good  
    vtss\_phy\_statistic\_t, 187  
sflow\_port\_no  
    vtss\_packet\_rx\_info\_t, 139  
sflow\_queue  
    vtss\_packet\_rx\_queue\_map\_t, 149  
sflow\_type  
    vtss\_packet\_rx\_info\_t, 139  
sfp\_dac  
    vtss\_port\_serd़es\_conf\_t, 226  
sgmii\_in\_pre  
    vtss\_phy\_mac\_serд\_pcs\_cntl\_t, 178  
sgmii\_out\_pre  
    vtss\_phy\_mac\_serд\_pcs\_cntl\_t, 178  
shaper\_port  
    vtss\_qos\_port\_conf\_t, 252  
shaper\_queue  
    vtss\_qos\_port\_conf\_t, 252  
sigdet  
    vtss\_phy\_conf\_t, 168  
sip  
    vtss\_ace\_frame\_arp\_t, 29  
    vtss\_ace\_frame\_ipv4\_t, 33  
    vtss\_ace\_frame\_ipv6\_t, 37  
    vtss\_ace\_sip\_smac\_t, 45  
    vtss\_ece\_frame\_ipv4\_t, 71  
    vtss\_ece\_frame\_ipv6\_t, 72  
    vtss\_qce\_frame\_ipv4\_t, 235  
    vtss\_qce\_frame\_ipv6\_t, 237  
    vtss\_vce\_frame\_ipv4\_t, 295  
    vtss\_vce\_frame\_ipv6\_t, 297  
sip\_dip\_enable  
    vtss\_aggr\_mode\_t, 57  
sip\_eq\_dip  
    vtss\_ace\_frame\_ipv4\_t, 35  
    vtss\_ace\_frame\_ipv6\_t, 40  
sip\_smac  
    vtss\_ace\_frame\_ipv4\_t, 36  
size  
    vtss\_packet\_rx\_queue\_conf\_t, 146  
skip\_coma  
    vtss\_phy\_conf\_t, 169  
smac  
    vtss\_ace\_frame\_arp\_t, 28  
                vtss\_ace\_frame\_etype\_t, 31  
                vtss\_ace\_frame\_llc\_t, 41  
                vtss\_ace\_frame\_snap\_t, 43  
                vtss\_ace\_sip\_smac\_t, 45  
                vtss\_ece\_mac\_t, 76  
                vtss\_port\_flow\_control\_conf\_t, 211  
                vtss\_qce\_mac\_t, 243  
                vtss\_vce\_mac\_t, 302  
smac\_enable  
    vtss\_aggr\_mode\_t, 56  
smac\_match  
    vtss\_ace\_frame\_arp\_t, 28  
snap  
    vtss\_ace\_frame\_snap\_t, 43  
    vtss\_ace\_t, 48  
    vtss\_qce\_key\_t, 241  
    vtss\_vce\_key\_t, 300  
speed  
    port\_custom\_conf\_t, 24  
    vtss\_phy\_forced\_t, 172  
    vtss\_port\_conf\_t, 203  
    vtss\_port\_status\_t, 229  
speed\_100M  
    vtss\_port\_sgmii\_aneg\_t, 227  
speed\_100m\_fdx  
    vtss\_phy\_aneg\_t, 163  
speed\_100m\_hdx  
    vtss\_phy\_aneg\_t, 163  
speed\_10M  
    vtss\_port\_sgmii\_aneg\_t, 227  
speed\_10m\_fdx  
    vtss\_phy\_aneg\_t, 163  
speed\_10m\_hdx  
    vtss\_phy\_aneg\_t, 163  
speed\_1G  
    vtss\_port\_sgmii\_aneg\_t, 227  
speed\_1g\_fdx  
    vtss\_phy\_aneg\_t, 163  
speed\_1g\_hdx  
    vtss\_phy\_aneg\_t, 163  
spi\_32bit\_read\_write  
    vtss\_init\_conf\_t, 95  
spi\_64bit\_read\_write  
    vtss\_init\_conf\_t, 95  
spi\_read\_write  
    vtss\_init\_conf\_t, 95  
sport  
    vtss\_ace\_frame\_ipv4\_t, 34  
    vtss\_ace\_frame\_ipv6\_t, 38  
    vtss\_ece\_frame\_ipv4\_t, 71  
    vtss\_ece\_frame\_ipv6\_t, 73  
    vtss\_qce\_frame\_ipv4\_t, 236  
    vtss\_qce\_frame\_ipv6\_t, 237  
sport\_dport\_enable  
    vtss\_aggr\_mode\_t, 57  
sport\_eq\_dport  
    vtss\_ace\_frame\_ipv4\_t, 36  
    vtss\_ace\_frame\_ipv6\_t, 40

squelch  
     vtss\_phy\_clock\_conf\_t, 165

squelsh  
     vtss\_sync\_clock\_in\_t, 268

src  
     vtss\_phy\_clock\_conf\_t, 165

stack\_queue  
     vtss\_packet\_rx\_queue\_map\_t, 149

status  
     vtss\_phy\_veriphy\_result\_t, 192

stripped\_tag  
     vtss\_packet\_rx\_info\_t, 135

sw\_tstamp  
     vtss\_packet\_rx\_info\_t, 137  
     vtss\_packet\_rx\_meta\_t, 144

switch\_frm  
     vtss\_packet\_tx\_info\_t, 154

symmetric\_pause  
     vtss\_phy\_aneg\_t, 164  
     vtss\_port\_clause\_37\_adv\_t, 199

TRUE  
     types.h, 356

tag  
     vtss\_ece\_key\_t, 74  
     vtss\_packet\_rx\_header\_t, 131  
     vtss\_packet\_rx\_info\_t, 135  
     vtss\_packet\_tx\_info\_t, 155  
     vtss\_qce\_key\_t, 240  
     vtss\_vce\_key\_t, 299

tag\_class\_enable  
     vtss\_qos\_port\_conf\_t, 253

tag\_default\_dei  
     vtss\_qos\_port\_conf\_t, 255

tag\_default\_pcp  
     vtss\_qos\_port\_conf\_t, 255

tag\_dei\_map  
     vtss\_qos\_port\_conf\_t, 255

tag\_pcp\_map  
     vtss\_qos\_port\_conf\_t, 255

tag\_remark\_mode  
     vtss\_qos\_port\_conf\_t, 255

tag\_type  
     vtss\_packet\_rx\_info\_t, 134

tagged  
     vtss\_ace\_vlan\_t, 50  
     vtss\_ece\_tag\_t, 80  
     vtss\_qce\_tag\_t, 245  
     vtss\_vce\_tag\_t, 304

tagprio  
     vtss\_tci\_t, 271

target  
     vtss\_inst\_create\_t, 97

tbi  
     vtss\_phy\_reset\_conf\_t, 184

tcp\_ack  
     vtss\_ace\_frame\_ipv4\_t, 35  
     vtss\_ace\_frame\_ipv6\_t, 39

tcp\_fin  
     vtss\_ace\_frame\_ipv4\_t, 34  
     vtss\_ace\_frame\_ipv6\_t, 39

tcp\_psh  
     vtss\_ace\_frame\_ipv4\_t, 35  
     vtss\_ace\_frame\_ipv6\_t, 39

tcp\_RST  
     vtss\_ace\_frame\_ipv4\_t, 35  
     vtss\_ace\_frame\_ipv6\_t, 39

tcp\_SYN  
     vtss\_ace\_frame\_ipv4\_t, 34  
     vtss\_ace\_frame\_ipv6\_t, 39

tcp\_URG  
     vtss\_ace\_frame\_ipv4\_t, 35  
     vtss\_ace\_frame\_ipv6\_t, 40

timeout  
     vtss\_mtimer\_t, 122

tod\_get\_ns\_cnt\_cb\_t  
     vtss\_misc\_api.h, 475

tpid  
     vtss\_packet\_port\_filter\_t, 127  
     vtss\_vlan\_tag\_t, 310

trans\_vid  
     vtss\_vlan\_trans\_grp2vlan\_conf\_t, 311

ts  
     vtss\_ts\_timestamp\_t, 279

ts\_id  
     vtss\_ts\_id\_t, 275

ts\_valid  
     vtss\_ts\_timestamp\_t, 279

tstamp\_id  
     vtss\_packet\_rx\_info\_t, 138

tstamp\_id\_decoded  
     vtss\_packet\_rx\_info\_t, 138

tt\_loop  
     vtss\_mce\_t, 119

ttl  
     vtss\_ace\_frame\_ipv4\_t, 32  
     vtss\_ace\_frame\_ipv6\_t, 38

tx\_clk\_skew\_ps  
     vtss\_phy\_rgmiil\_conf\_t, 186

tx\_etherStatsBroadcastPkts  
     vtss\_port\_rmon\_counters\_t, 223

tx\_etherStatsCollisions  
     vtss\_port\_rmon\_counters\_t, 224

tx\_etherStatsDropEvents  
     vtss\_port\_rmon\_counters\_t, 223

tx\_etherStatsMulticastPkts  
     vtss\_port\_rmon\_counters\_t, 223

tx\_etherStatsOctets  
     vtss\_port\_rmon\_counters\_t, 223

tx\_etherStatsPkts  
     vtss\_port\_rmon\_counters\_t, 223

tx\_etherStatsPkts1024to1518Octets  
     vtss\_port\_rmon\_counters\_t, 225

tx\_etherStatsPkts128to255Octets  
     vtss\_port\_rmon\_counters\_t, 224

tx\_etherStatsPkts1519toMaxOctets  
     vtss\_port\_rmon\_counters\_t, 225

tx\_etherStatsPkts256to511Octets  
    vtss\_port\_rmon\_counters\_t, 224

tx\_etherStatsPkts512to1023Octets  
    vtss\_port\_rmon\_counters\_t, 225

tx\_etherStatsPkts64Octets  
    vtss\_port\_rmon\_counters\_t, 224

tx\_etherStatsPkts65to127Octets  
    vtss\_port\_rmon\_counters\_t, 224

tx\_frames  
    vtss\_basic\_counters\_t, 60

tx\_green  
    vtss\_port\_evc\_counters\_t, 209

tx\_out\_bytes  
    vtss\_eee\_port\_counter\_t, 83

tx\_out\_bytes\_get  
    vtss\_eee\_port\_counter\_t, 83

tx\_prio  
    vtss\_port\_proprietary\_counters\_t, 218

tx\_remote\_fault  
    vtss\_phy\_aneg\_t, 164

tx\_tw  
    vtss\_eee\_port\_conf\_t, 81

tx\_yellow  
    vtss\_port\_evc\_counters\_t, 210

type  
    vtss\_ace\_t, 47  
    vtss\_dlb\_policer\_conf\_t, 66  
    vtss\_ece\_key\_t, 74  
    vtss\_eps\_port\_conf\_t, 85  
    vtss\_evc\_inner\_tag\_t, 87  
    vtss\_fan\_conf\_t, 92  
    vtss\_ip\_addr\_t, 99  
    vtss\_qce\_key\_t, 240  
    vtss\_routing\_entry\_t, 259  
    vtss\_sflow\_port\_conf\_t, 263  
    vtss\_vcap\_vr\_t, 291  
    vtss\_vce\_key\_t, 300

types.h  
    BOOL, 374  
    FALSE, 356  
    i16, 373  
    i32, 373  
    i64, 373  
    i8, 373  
    MAC\_ADDR\_BROADCAST, 364  
    PRIi64, 354  
    PRIlu64, 354  
    PRIx64, 354  
    TRUE, 356  
    u16, 374  
    u32, 374  
    u64, 374  
    u8, 373  
    uintptr\_t, 374  
    VTSS\_ACL\_POLICER\_NO\_END, 368  
    VTSS\_ACL\_POLICER\_NO\_START, 368  
    VTSS\_ACL\_POLICERS, 368  
    VTSS\_ACL\_POLICIES, 369

    VTSS\_ACL\_POLICY\_NO\_END, 370  
    VTSS\_ACL\_POLICY\_NO\_MAX, 369  
    VTSS\_ACL\_POLICY\_NO\_MIN, 369  
    VTSS\_ACL\_POLICY\_NO\_NONE, 369  
    VTSS\_ACL\_POLICY\_NO\_START, 369  
    VTSS\_AGGR\_NO\_END, 365  
    VTSS\_AGGR\_NO\_NONE, 365  
    VTSS\_AGGR\_NO\_START, 365  
    VTSS\_AGGRS, 364  
    VTSS\_BIT64, 355  
    VTSS\_BITMASK64, 355  
    VTSS\_BITRATE\_DISABLED, 362  
    VTSS\_CLOCK\_IDENTITY\_LENGTH, 372  
    VTSS\_DEI\_ARRAY\_SIZE, 361  
    VTSS\_DEI\_END, 361  
    VTSS\_DEI\_START, 361  
    VTSS\_DEIS, 361  
    VTSS\_DPL\_ARRAY\_SIZE, 362  
    VTSS\_DPL\_END, 362  
    VTSS\_DPL\_START, 362  
    VTSS\_DPLS, 361  
    VTSS\_ENCODE\_BITFIELD64, 355  
    VTSS\_ENCODE\_BITMASK64, 355  
    VTSSETYPE\_VTSS, 363  
    VTSS\_EVCS, 364  
    VTSS\_EXTRACT\_BITFIELD64, 355  
    VTSS\_GLAG\_NO\_END, 366  
    VTSS\_GLAG\_NO\_NONE, 365  
    VTSS\_GLAG\_NO\_START, 366  
    VTSS\_GLAG\_PORT\_ARRAY\_SIZE, 367  
    VTSS\_GLAG\_PORT\_END, 366  
    VTSS\_GLAG\_PORT\_START, 366  
    VTSS\_GLAG\_PORTS, 366  
    VTSS\_GLAGS, 365  
    VTSS\_HQOS\_COUNT, 370  
    VTSS\_HQOS\_ID\_NONE, 370  
    VTSS\_INTERVAL\_MS, 371  
    VTSS\_INTERVAL\_NS, 371  
    VTSS\_INTERVAL\_PS, 372  
    VTSS\_INTERVAL\_SEC, 371  
    VTSS\_INTERVAL\_US, 371  
    VTSS\_ISDX\_NONE, 364  
    VTSS\_MAC\_ADDR\_SZ\_BYTES, 364  
    VTSS\_MAX\_TIMEINTERVAL, 371  
    VTSS\_ONE\_MILL, 370  
    VTSS\_ONE\_MIA, 370  
    VTSS\_PACKET\_RATE\_DISABLED, 356  
    VTSS\_PACKET\_RX\_GRP\_CNT, 367  
    VTSS\_PACKET\_RX\_QUEUE\_CNT, 367  
    VTSS\_PACKET\_RX\_QUEUE\_END, 368  
    VTSS\_PACKET\_RX\_QUEUE\_NONE, 367  
    VTSS\_PACKET\_RX\_QUEUE\_START, 368  
    VTSS\_PACKET\_TX\_GRP\_CNT, 367  
    VTSS\_PCP\_ARRAY\_SIZE, 360  
    VTSS\_PCP\_END, 360  
    VTSS\_PCP\_START, 360  
    VTSS\_PCPS, 360  
    VTSS\_PORT\_ARRAY\_SIZE, 358

VTSS\_PORT\_COUNT, 356  
 VTSS\_PORT\_IS\_PORT, 358  
 VTSS\_PORT\_NO\_CPU, 357  
 VTSS\_PORT\_NO\_END, 357  
 VTSS\_PORT\_NO\_NONE, 357  
 VTSS\_PORT\_NO\_START, 357  
 VTSS\_PORTS, 357  
 VTSS\_PRIO\_ARRAY\_SIZE, 359  
 VTSS\_PRIO\_END, 359  
 VTSS\_PRIO\_NO\_NONE, 358  
 VTSS\_PRIO\_START, 358  
 VTSS\_PRIOS, 358  
 VTSS\_QUEUE\_ARRAY\_SIZE, 360  
 VTSS\_QUEUE\_END, 359  
 VTSS\_QUEUE\_START, 359  
 VTSS\_QUEUES, 359  
 VTSS\_SEC\_NS\_INTERVAL, 372  
 VTSS\_SYNCE\_CLK\_PORT\_ARRAY\_SIZE, 372  
 VTSS\_VID\_ALL, 363  
 VTSS\_VID\_DEFAULT, 363  
 VTSS\_VID\_NULL, 362  
 VTSS\_VID\_RESERVED, 363  
 VTSS\_VIDS, 363  
 vtss\_ece\_dir\_t, 382  
 vtss\_ece\_pop\_tag\_t, 383  
 vtss\_hqos\_sch\_mode\_t, 383  
 vtss\_ip\_type\_t, 381  
 vtss\_isdx\_t, 375  
 vtss\_mac\_addr\_t, 375  
 vtss\_mem\_flags\_t, 378  
 vtss\_packet\_reg\_type\_t, 380  
 vtss\_packet\_rx\_grp\_t, 375  
 vtss\_packet\_tx\_grp\_t, 375  
 vtss\_policer\_type\_t, 380  
 vtss\_port\_interface\_t, 378  
 vtss\_serd़es\_mode\_t, 379  
 vtss\_storm\_policer\_mode\_t, 380  
 vtss\_vcap\_bit\_t, 381  
 vtss\_vcap\_key\_type\_t, 382  
 vtss\_vcap\_vr\_type\_t, 382  
 vtss\_vdd\_t, 381  
 vtss\_vlan\_frame\_t, 380

u16  
 types.h, 374

u32  
 types.h, 374

u64  
 types.h, 374

u8  
 types.h, 373

uint  
 vtss\_os\_custom.h, 505

uintptr\_t  
 types.h, 374

ulong  
 vtss\_os\_custom.h, 505

unidir  
 vtss\_phy\_conf\_t, 168

unidirectional\_ability  
 vtss\_port\_status\_t, 230

unknown  
 vtss\_ace\_frame\_arp\_t, 28

untagged\_vid  
 vtss\_vlan\_port\_conf\_t, 308

use\_extended\_bus\_cycle  
 vtss\_pi\_conf\_t, 195

usr\_prio  
 vtss\_ace\_vlan\_t, 50  
 vtss\_qos\_port\_conf\_t, 253

uvid  
 vtss\_evc\_pb\_conf\_t, 89

v  
 vtss\_vcap\_vr\_t, 291

VTSS\_ACE\_ID\_LAST  
 vtss\_security\_api.h, 650

VTSS\_ACL\_POLICER\_NO\_END  
 types.h, 368

VTSS\_ACL\_POLICER\_NO\_START  
 types.h, 368

VTSS\_ACL\_POLICERS  
 types.h, 368

VTSS\_ACL\_POLICIES  
 types.h, 369

VTSS\_ACL\_POLICY\_NO\_END  
 types.h, 370

VTSS\_ACL\_POLICY\_NO\_MAX  
 types.h, 369

VTSS\_ACL\_POLICY\_NO\_MIN  
 types.h, 369

VTSS\_ACL\_POLICY\_NO\_NONE  
 types.h, 369

VTSS\_ACL\_POLICY\_NO\_START  
 types.h, 369

VTSS\_AGGR\_NO\_END  
 types.h, 365

VTSS\_AGGR\_NO\_NONE  
 types.h, 365

VTSS\_AGGR\_NO\_START  
 types.h, 365

VTSS\_AGGRS  
 types.h, 364

VTSS\_ARCH\_CARACAL  
 options.h, 317

VTSS\_ARCH\_LUTON26  
 options.h, 318

VTSS\_BIT64  
 types.h, 355

VTSS\_BITMASK64  
 types.h, 355

VTSS\_BITRATE\_DISABLED  
 types.h, 362

VTSS\_CHIP CU PHY  
 options.h, 331

VTSS\_CLOCK\_IDENTITY\_LENGTH  
 types.h, 372

VTSS\_DEI\_ARRAY\_SIZE

types.h, 361  
VTSS\_DEI\_END  
types.h, 361  
VTSS\_DEI\_START  
types.h, 361  
VTSS\_DEIS  
types.h, 361  
VTSS\_DIV64  
vtss\_os\_custom.h, 506  
vtss\_os\_ecos.h, 511  
vtss\_os\_linux.h, 523  
VTSS\_DPL\_ARRAY\_SIZE  
types.h, 362  
VTSS\_DPL\_END  
types.h, 362  
VTSS\_DPL\_START  
types.h, 362  
VTSS\_DPLS  
types.h, 361  
VTSS\_ECE\_ID\_LAST  
vtss\_evc\_api.h, 388  
VTSS\_ENCODE\_BITFIELD64  
types.h, 355  
VTSS\_ENCODE\_BITMASK64  
types.h, 355  
VTSS\_ERPI\_ARRAY\_SIZE  
vtss\_l2\_api.h, 424  
VTSS\_ERPI\_END  
vtss\_l2\_api.h, 424  
VTSS\_ERPI\_START  
vtss\_l2\_api.h, 423  
VTSS\_ERPIS  
vtss\_l2\_api.h, 423  
VTSS\_ETYPE\_VTSS  
types.h, 363  
VTSS\_EVC\_ID\_NONE  
vtss\_evc\_api.h, 387  
VTSS\_EVC\_POLICERS  
vtss\_evc\_api.h, 387  
VTSS\_EVCS  
types.h, 364  
VTSS\_EXTRACT\_BITFIELD64  
types.h, 355  
VTSS\_FEATURE\_ACL\_V2  
options.h, 328  
VTSS\_FEATURE\_ACL  
options.h, 328  
VTSS\_FEATURE\_CLAUSE\_37  
options.h, 319  
VTSS\_FEATURE\_EEE  
options.h, 327  
VTSS\_FEATURE\_EVC  
options.h, 318  
VTSS\_FEATURE\_EXC\_COL\_CONT  
options.h, 319  
VTSS\_FEATURE\_FAN  
options.h, 327  
VTSS\_FEATURE\_INTERRUPTS  
options.h, 330  
VTSS FEATURE IPV4 MC SIP  
options.h, 326  
VTSS FEATURE IPV6 MC SIP  
options.h, 326  
VTSS FEATURE IRQ CONTROL  
options.h, 329  
VTSS FEATURE LAYER2  
options.h, 325  
VTSS FEATURE LED POW REDUC  
options.h, 329  
VTSS FEATURE MAC AGE AUTO  
options.h, 326  
VTSS FEATURE MAC CPU QUEUE  
options.h, 327  
VTSS FEATURE MIRROR CPU  
options.h, 330  
VTSS FEATURE\_MISC  
options.h, 318  
VTSS FEATURE\_NPI  
options.h, 329  
VTSS FEATURE\_PACKET\_GROUPING  
options.h, 325  
VTSS FEATURE\_PACKET\_PORT\_REG  
options.h, 325  
VTSS FEATURE\_PACKET\_RX  
options.h, 325  
VTSS FEATURE\_PACKET\_TX  
options.h, 324  
VTSS FEATURE\_PACKET  
options.h, 324  
VTSS FEATURE\_PORT\_CNT\_BRIDGE  
options.h, 319  
VTSS FEATURE\_PORT\_CONTROL  
options.h, 319  
VTSS FEATURE\_PORT\_MUX  
options.h, 327  
VTSS FEATURE\_PVLAN  
options.h, 325  
VTSS FEATURE\_QCL\_DMAC\_DIP  
options.h, 320  
VTSS FEATURE\_QCL\_KEY\_S\_TAG  
options.h, 320  
VTSS FEATURE\_QCL\_PCP\_DEI\_ACTION  
options.h, 320  
VTSS FEATURE\_QCL\_POLICY\_ACTION  
options.h, 321  
VTSS FEATURE\_QCL\_V2  
options.h, 320  
VTSS FEATURE\_QCL  
options.h, 320  
VTSS FEATURE\_QOS\_CLASSIFICATION\_V2  
options.h, 323  
VTSS FEATURE\_QOS\_DSCP\_CLASS\_DP\_AWARE  
options.h, 323  
VTSS FEATURE\_QOS\_DSCP\_REMARK\_DP\_AWA↔  
RE  
options.h, 324

VTSS\_FEATURE\_QOS\_DSCP\_REMARK\_V2  
options.h, 324

VTSS\_FEATURE\_QOS\_DSCP\_REMARK  
options.h, 324

VTSS\_FEATURE\_QOS\_EGRESS\_QUEUE\_SHAPE\_RS\_EB  
options.h, 323

VTSS\_FEATURE\_QOS\_EGRESS\_QUEUE\_SHAPE\_RS  
options.h, 323

VTSS\_FEATURE\_QOS\_POLICER\_BC\_SWITCH  
options.h, 321

VTSS\_FEATURE\_QOS\_POLICER\_DLBB  
options.h, 318

VTSS\_FEATURE\_QOS\_POLICER\_MC\_SWITCH  
options.h, 321

VTSS\_FEATURE\_QOS\_POLICER\_UC\_SWITCH  
options.h, 321

VTSS\_FEATURE\_QOS\_PORT\_POLICER\_EXT\_FPS  
options.h, 322

VTSS\_FEATURE\_QOS\_PORT\_POLICER\_EXT\_FC  
options.h, 322

VTSS\_FEATURE\_QOS\_PORT\_POLICER\_EXT  
options.h, 321

VTSS\_FEATURE\_QOS\_QUEUE\_POLICER  
options.h, 322

VTSS\_FEATURE\_QOS\_QUEUE\_TX  
options.h, 322

VTSS\_FEATURE\_QOS\_SCHEDULER\_V2  
options.h, 322

VTSS\_FEATURE\_QOS\_TAG\_REMARK\_V2  
options.h, 323

VTSS\_FEATURE\_QOS  
options.h, 319

VTSS\_FEATURE\_SERDES\_MACRO\_SETTINGS  
options.h, 330

VTSS\_FEATURE\_SERIAL\_GPIO  
options.h, 318

VTSS\_FEATURE\_SFLOW  
options.h, 330

VTSS\_FEATURE\_SYNCE  
options.h, 329

VTSS\_FEATURE\_TIMESTAMP\_LATENCY\_COMP  
options.h, 329

VTSS\_FEATURE\_TIMESTAMP\_ONE\_STEP  
options.h, 328

VTSS\_FEATURE\_TIMESTAMP  
options.h, 328

VTSS\_FEATURE\_VCAP  
options.h, 327, 332

VTSS\_FEATURE\_VCL  
options.h, 328

VTSS\_FEATURE\_VLAN\_PORT\_V2  
options.h, 326

VTSS\_FEATURE\_VLAN\_TRANSLATION  
options.h, 330

VTSS\_FEATURE\_VLAN\_TX\_TAG  
options.h, 326

VTSS\_FEATURE\_WARM\_START  
options.h, 331

VTSS\_FRAME\_GAP\_DEFAULT  
vtss\_port\_api.h, 628

VTSS\_GLAG\_NO\_END  
types.h, 366

VTSS\_GLAG\_NO\_NONE  
types.h, 365

VTSS\_GLAG\_NO\_START  
types.h, 366

VTSS\_GLAG\_PORT\_ARRAY\_SIZE  
types.h, 367

VTSS\_GLAG\_PORT\_END  
types.h, 366

VTSS\_GLAG\_PORT\_START  
types.h, 366

VTSS\_GLAG\_PORTS  
types.h, 366

VTSS\_GLAGS  
types.h, 365

VTSS\_HQOS\_COUNT  
types.h, 370

VTSS\_HQOS\_ID\_NONE  
types.h, 370

VTSS\_I2C\_NO\_MULTIPLEXER  
vtss\_init\_api.h, 399

VTSS\_INTERVAL\_MS  
types.h, 371

VTSS\_INTERVAL\_NS  
types.h, 371

VTSS\_INTERVAL\_PS  
types.h, 372

VTSS\_INTERVAL\_SEC  
types.h, 371

VTSS\_INTERVAL\_US  
types.h, 371

VTSS\_ISDX\_NONE  
types.h, 364

VTSS\_JR1\_PACKET\_HDR\_SIZE\_BYTES  
vtss\_packet\_api.h, 530

VTSS\_JR1\_RX\_IFH\_SIZE  
vtss\_packet\_api.h, 531

VTSS\_JR2\_PACKET\_HDR\_SIZE\_BYTES  
vtss\_packet\_api.h, 530

VTSS\_JR2\_RX\_IFH\_SIZE  
vtss\_packet\_api.h, 531

VTSS\_L26\_PACKET\_HDR\_SIZE\_BYTES  
vtss\_packet\_api.h, 530

VTSS\_L26\_RX\_IFH\_SIZE  
vtss\_packet\_api.h, 531

VTSS\_LABS  
vtss\_os\_custom.h, 506

vtss\_os\_ecos.h, 512

vtss\_os\_linux.h, 523

VTSS\_LLabs  
vtss\_os\_custom.h, 507

vtss\_os\_ecos.h, 512

vtss\_os\_linux.h, 523

VTSS\_MAC\_ADDR\_SZ\_BYTES  
types.h, 364

VTSS\_MAC\_ADDRS  
vtss\_l2\_api.h, 418

VTSS\_MAX\_FRAME\_LENGTH\_MAX  
vtss\_port\_api.h, 629

VTSS\_MAX\_FRAME\_LENGTH\_STANDARD  
vtss\_port\_api.h, 628

VTSS\_MAX\_TIMEINTERVAL  
types.h, 371

VTSS\_MCE\_ID\_LAST  
vtss\_evc\_api.h, 388

VTSS\_MCE\_POP\_NONE  
vtss\_evc\_api.h, 388

VTSS\_MOD64  
vtss\_os\_custom.h, 506  
vtss\_os\_ecos.h, 511  
vtss\_os\_linux.h, 523

VTSS\_MSLEEP  
vtss\_os\_custom.h, 505  
vtss\_os\_ecos.h, 510  
vtss\_os\_linux.h, 520

VTSS\_MSTI\_ARRAY\_SIZE  
vtss\_l2\_api.h, 418

VTSS\_MSTI\_END  
vtss\_l2\_api.h, 418

VTSS\_MSTI\_START  
vtss\_l2\_api.h, 418

VTSS\_MSTIS  
vtss\_l2\_api.h, 418

VTSS\_MTIMER\_CANCEL  
vtss\_os\_custom.h, 506  
vtss\_os\_ecos.h, 511  
vtss\_os\_linux.h, 521

VTSS\_MTIMER\_START  
vtss\_os\_custom.h, 505  
vtss\_os\_ecos.h, 510  
vtss\_os\_linux.h, 521

VTSS\_MTIMER\_TIMEOUT  
vtss\_os\_custom.h, 505  
vtss\_os\_ecos.h, 511  
vtss\_os\_linux.h, 521

VTSS\_NSLEEP  
vtss\_os\_ecos.h, 510  
vtss\_os\_linux.h, 520

VTSS\_ONE\_MILL  
types.h, 370

VTSS\_ONE\_MIA  
types.h, 370

VTSS\_OPT\_TRACE  
options.h, 331

VTSS\_OPT\_VAUI\_EQ\_CTRL  
options.h, 331

VTSS\_OS\_BIG\_ENDIAN  
vtss\_os\_ecos.h, 515  
vtss\_os\_linux.h, 520

VTSS\_OS\_COMPILER\_ATTRIBUTE\_ALIGNED  
vtss\_os\_ecos.h, 514

VTSS\_OS\_CTZ64  
vtss\_os\_custom.h, 507  
vtss\_os\_ecos.h, 512  
vtss\_os\_linux.h, 525

VTSS\_OS\_CTZ  
vtss\_os\_custom.h, 507  
vtss\_os\_ecos.h, 512  
vtss\_os\_linux.h, 524

VTSS\_OS\_DCACHE\_FLUSH  
vtss\_os\_ecos.h, 515

VTSS\_OS\_DCACHE\_INVALIDATE  
vtss\_os\_ecos.h, 514

VTSS\_OS\_DCACHE\_LINE\_SIZE\_BYTES  
vtss\_os\_ecos.h, 514

VTSS\_OS\_FREE  
vtss\_os\_custom.h, 508  
vtss\_os\_ecos.h, 513  
vtss\_os\_linux.h, 525

VTSS\_OS\_INTERRUPT\_DISABLE  
vtss\_os\_ecos.h, 517

VTSS\_OS\_INTERRUPT\_FLAGS  
vtss\_os\_ecos.h, 516

VTSS\_OS\_INTERRUPT\_RESTORE  
vtss\_os\_ecos.h, 517

VTSS\_OS\_MALLOC  
vtss\_os\_custom.h, 507  
vtss\_os\_ecos.h, 513  
vtss\_os\_linux.h, 525

VTSS\_OS\_NTOHL  
vtss\_os\_ecos.h, 515  
vtss\_os\_linux.h, 520

VTSS\_OS\_RAND  
vtss\_os\_custom.h, 508  
vtss\_os\_ecos.h, 513  
vtss\_os\_linux.h, 526

VTSS\_OS\_REORDER\_BARRIER  
vtss\_os\_ecos.h, 514

VTSS\_OS\_SCHEDULER\_FLAGS  
vtss\_os\_ecos.h, 516  
vtss\_os\_linux.h, 522

VTSS\_OS\_SCHEDULER\_LOCK  
vtss\_os\_ecos.h, 516  
vtss\_os\_linux.h, 522

VTSS\_OS\_SCHEDULER\_UNLOCK  
vtss\_os\_ecos.h, 516  
vtss\_os\_linux.h, 522

VTSS\_OS\_TIMESTAMP\_TYPE  
vtss\_misc\_api.h, 474

VTSS\_OS\_TIMESTAMP  
vtss\_misc\_api.h, 474

VTSS\_OS\_VIRT\_TO\_PHYS  
vtss\_os\_ecos.h, 515

VTSS\_PACKET\_HDR\_SIZE\_BYTES  
vtss\_packet\_api.h, 530

VTSS\_PACKET\_RATE\_DISABLED  
types.h, 356

VTSS\_PACKET\_RX\_GRP\_CNT  
types.h, 367

VTSS\_PACKET\_RX\_QUEUE\_CNT  
types.h, 367

VTSS\_PACKET\_RX\_QUEUE\_END  
types.h, 368

VTSS\_PACKET\_RX\_QUEUE\_NONE  
types.h, 367

VTSS\_PACKET\_RX\_QUEUE\_START  
types.h, 368

VTSS\_PACKET\_TX\_GRP\_CNT  
types.h, 367

VTSS\_PACKET\_TX\_IFH\_MAX  
vtss\_packet\_api.h, 531

VTSS\_PCP\_ARRAY\_SIZE  
types.h, 360

VTSS\_PCP\_END  
types.h, 360

VTSS\_PCP\_START  
types.h, 360

VTSS\_PCPS  
types.h, 360

VTSS\_PHY\_ACTIPHY\_PWR  
vtss\_phy\_api.h, 559

VTSS\_PHY\_LINK\_AMS\_EV  
vtss\_phy\_api.h, 564

VTSS\_PHY\_LINK\_AUTO\_NEG\_COMPLETE\_EV  
vtss\_phy\_api.h, 564

VTSS\_PHY\_LINK\_AUTO\_NEG\_ERROR\_EV  
vtss\_phy\_api.h, 564

VTSS\_PHY\_LINK\_DOWN\_PWR  
vtss\_phy\_api.h, 559

VTSS\_PHY\_LINK\_EXT\_EEE\_LINKFAIL\_EV  
vtss\_phy\_api.h, 567

VTSS\_PHY\_LINK\_EXT\_EEE\_WAIT\_RX\_TQ\_EV  
vtss\_phy\_api.h, 567

VTSS\_PHY\_LINK\_EXT\_EEE\_WAIT\_TS\_EV  
vtss\_phy\_api.h, 567

VTSS\_PHY\_LINK\_EXT\_EEE\_WAKE\_ERR\_EV  
vtss\_phy\_api.h, 567

VTSS\_PHY\_LINK\_EXT\_MACSEC\_EGRESS\_EV  
vtss\_phy\_api.h, 568

VTSS\_PHY\_LINK\_EXT\_MACSEC\_FC\_BUFF\_EV  
vtss\_phy\_api.h, 568

VTSS\_PHY\_LINK\_EXT\_MACSEC\_HOST\_MAC\_EV  
vtss\_phy\_api.h, 568

VTSS\_PHY\_LINK\_EXT\_MACSEC\_INGRESS\_EV  
vtss\_phy\_api.h, 568

VTSS\_PHY\_LINK\_EXT\_MACSEC\_LINE\_MAC\_EV  
vtss\_phy\_api.h, 568

VTSS\_PHY\_LINK\_EXT\_MEM\_INT\_RING\_EV  
vtss\_phy\_api.h, 569

VTSS\_PHY\_LINK\_EXT\_RR\_SW\_COMPL\_EV  
vtss\_phy\_api.h, 567

VTSS\_PHY\_LINK\_EXTENDED\_REG\_INT\_EV  
vtss\_phy\_api.h, 566

VTSS\_PHY\_LINK\_FALSE\_CARRIER\_INT\_EV  
vtss\_phy\_api.h, 565

VTSS\_PHY\_LINK\_FDX\_STATE\_CHANGE\_EV  
vtss\_phy\_api.h, 564

VTSS\_PHY\_LINK\_FFAIL\_EV  
vtss\_phy\_api.h, 563

VTSS\_PHY\_LINK\_INLINE\_POW\_DEV\_DETECT\_EV  
vtss\_phy\_api.h, 565

VTSS\_PHY\_LINK\_LINK\_SPEED\_DS\_DETECT\_EV  
vtss\_phy\_api.h, 566

VTSS\_PHY\_LINK\_LOS\_EV  
vtss\_phy\_api.h, 563

VTSS\_PHY\_LINK\_MASTER\_SLAVE\_RES\_ERR\_EV  
vtss\_phy\_api.h, 566

VTSS\_PHY\_LINK\_RX\_ER\_INT\_EV  
vtss\_phy\_api.h, 566

VTSS\_PHY\_LINK\_RX\_FIFO\_OVERFLOW\_INT\_EV  
vtss\_phy\_api.h, 565

VTSS\_PHY\_LINK\_SPEED\_STATE\_CHANGE\_EV  
vtss\_phy\_api.h, 564

VTSS\_PHY\_LINK\_SYMBOL\_ERR\_INT\_EV  
vtss\_phy\_api.h, 565

VTSS\_PHY\_LINK\_TX\_FIFO\_OVERFLOW\_INT\_EV  
vtss\_phy\_api.h, 565

VTSS\_PHY\_LINK\_UP\_FULL\_PWR  
vtss\_phy\_api.h, 559

VTSS\_PHY\_LINK\_WAKE\_ON\_LAN\_INT\_EV  
vtss\_phy\_api.h, 566

VTSS\_PHY\_OPT\_VERIPHYS  
options.h, 331

VTSS\_PHY\_PAGE\_0x2DAF  
vtss\_phy\_api.h, 562

VTSS\_PHY\_PAGE\_1588  
vtss\_phy\_api.h, 561

VTSS\_PHY\_PAGE\_EXTENDED\_2  
vtss\_phy\_api.h, 560

VTSS\_PHY\_PAGE\_EXTENDED\_3  
vtss\_phy\_api.h, 561

VTSS\_PHY\_PAGE\_EXTENDED\_4  
vtss\_phy\_api.h, 561

VTSS\_PHY\_PAGE\_EXTENDED  
vtss\_phy\_api.h, 560

VTSS\_PHY\_PAGE\_GPIO  
vtss\_phy\_api.h, 561

VTSS\_PHY\_PAGE\_MACSEC  
vtss\_phy\_api.h, 561

VTSS\_PHY\_PAGE\_STANDARD  
vtss\_phy\_api.h, 560

VTSS\_PHY\_PAGE\_TEST  
vtss\_phy\_api.h, 562

VTSS\_PHY\_PAGE\_TR  
vtss\_phy\_api.h, 562

VTSS\_PHY\_POWER\_ACTIPHY\_BIT  
phy.h, 333  
vtss\_phy\_api.h, 558

VTSS\_PHY\_POWER\_DYNAMIC\_BIT  
phy.h, 333  
vtss\_phy\_api.h, 559

VTSS\_PHY\_RECov\_CLK1  
vtss\_phy\_api.h, 559

VTSS\_PHY\_RECov\_CLK2  
vtss\_phy\_api.h, 560

VTSS\_PHY\_RECov\_CLK\_NUM  
vtss\_phy\_api.h, 560

VTSS\_PHY\_REG\_EXTENDED  
vtss\_phy\_api.h, 562

VTSS\_PHY\_REG\_GPIO  
vtss\_phy\_api.h, 563

VTSS\_PHY\_REG\_STANDARD  
vtss\_phy\_api.h, 562

VTSS\_PHY\_REG\_TEST  
vtss\_phy\_api.h, 563

VTSS\_PHY\_REG\_TR  
vtss\_phy\_api.h, 563

VTSS\_PORT\_ARRAY\_SIZE  
types.h, 358

VTSS\_PORT\_COUNT  
types.h, 356

VTSS\_PORT\_IS\_PORT  
types.h, 358

VTSS\_PORT\_NO\_CPU  
types.h, 357

VTSS\_PORT\_NO\_END  
types.h, 357

VTSS\_PORT\_NO\_NONE  
types.h, 357

VTSS\_PORT\_NO\_START  
types.h, 357

VTSS\_PORT\_POLICER\_CPU\_QUEUES  
vtss\_qos\_api.h, 640

VTSS\_PORT\_POLICERS  
vtss\_qos\_api.h, 640

VTSS\_PORTS  
types.h, 357

VTSS\_PRIO\_ARRAY\_SIZE  
types.h, 359

VTSS\_PRIO\_END  
types.h, 359

VTSS\_PRIO\_NO\_NONE  
types.h, 358

VTSS\_PRIO\_START  
types.h, 358

VTSS\_PRIO\_SUPER  
vtss\_packet\_api.h, 529

VTSS\_PRIOS  
types.h, 358

VTSS\_PVLAN\_ARRAY\_SIZE  
vtss\_l2\_api.h, 423

VTSS\_PVLAN\_NO\_DEFAULT  
vtss\_l2\_api.h, 423

VTSS\_PVLAN\_NO\_END  
vtss\_l2\_api.h, 423

VTSS\_PVLAN\_NO\_START  
vtss\_l2\_api.h, 422

VTSS\_PVLANS  
vtss\_l2\_api.h, 422

VTSS\_QCE\_ID\_LAST  
vtss\_qos\_api.h, 641

VTSS\_QCL\_ARRAY\_SIZE  
vtss\_qos\_api.h, 641

VTSS\_QCL\_ID\_END  
vtss\_qos\_api.h, 641

VTSS\_QCL\_ID\_START  
vtss\_qos\_api.h, 641

VTSS\_QCL\_IDS  
vtss\_qos\_api.h, 641

VTSS\_QUEUE\_ARRAY\_SIZE  
types.h, 360

VTSS\_QUEUE\_END  
types.h, 359

VTSS\_QUEUE\_START  
types.h, 359

VTSS\_QUEUES  
types.h, 359

VTSS\_SEC\_NS\_INTERVAL  
types.h, 372

VTSS\_SVL\_PACKET\_HDR\_SIZE\_BYTES  
vtss\_packet\_api.h, 530

VTSS\_SVL\_RX\_IFH\_SIZE  
vtss\_packet\_api.h, 531

VTSS\_SYNCE\_CLK\_MAX  
vtss\_sync\_api.h, 659

VTSS\_SYNCE\_CLK\_PORT\_ARRAY\_SIZE  
types.h, 372

VTSS\_SYNCE\_CLK\_A  
vtss\_sync\_api.h, 659

VTSS\_SYNCE\_CLK\_B  
vtss\_sync\_api.h, 659

VTSS\_TIME\_OF\_DAY  
vtss\_os\_ecos.h, 511  
vtss\_os\_linux.h, 522

VTSS\_VCE\_ID\_LAST  
vtss\_l2\_api.h, 419

VTSS\_VCL\_ARRAY\_SIZE  
vtss\_l2\_api.h, 419

VTSS\_VCL\_ID\_END  
vtss\_l2\_api.h, 419

VTSS\_VCL\_ID\_START  
vtss\_l2\_api.h, 419

VTSS\_VCL\_IDS  
vtss\_l2\_api.h, 419

VTSS\_VID\_ALL  
types.h, 363

VTSS\_VID\_DEFAULT  
types.h, 363

VTSS\_VID\_NULL  
types.h, 362

VTSS\_VID\_RESERVED  
types.h, 363

VTSS\_VIDS  
types.h, 363

VTSS\_VLAN\_TRANS\_FIRST\_GROUP\_ID  
vtss\_l2\_api.h, 420

VTSS\_VLAN\_TRANS\_GROUP\_MAX\_CNT  
vtss\_l2\_api.h, 420

VTSS\_VLAN\_TRANS\_LAST\_GROUP\_ID  
vtss\_l2\_api.h, 421

VTSS\_VLAN\_TRANS\_MAX\_CNT

vtss\_l2\_api.h, 420  
 VTSS\_VLAN\_TRANS\_MAX\_VLAN\_ID  
   vtss\_l2\_api.h, 421  
 VTSS\_VLAN\_TRANS\_NULL\_CHECK  
   vtss\_l2\_api.h, 422  
 VTSS\_VLAN\_TRANS\_NULL\_GROUP\_ID  
   vtss\_l2\_api.h, 420  
 VTSS\_VLAN\_TRANS\_PORT\_BF\_SIZE  
   vtss\_l2\_api.h, 422  
 VTSS\_VLAN\_TRANS\_VALID\_GROUP\_CHECK  
   vtss\_l2\_api.h, 421  
 VTSS\_VLAN\_TRANS\_VALID\_VLAN\_CHECK  
   vtss\_l2\_api.h, 421  
 VTSS\_VLAN\_TRANS\_VID\_START  
   vtss\_l2\_api.h, 420  
 val  
   vtss\_eee\_port\_state\_t, 84  
   vtss\_phy\_conf\_1g\_t, 166  
 value  
   vtss\_sgpi\_port\_data\_t, 266  
   vtss\_vcap\_ip\_t, 280  
   vtss\_vcap\_u128\_t, 281  
   vtss\_vcap\_u16\_t, 282  
   vtss\_vcap\_u24\_t, 283  
   vtss\_vcap\_u32\_t, 284  
   vtss\_vcap\_u40\_t, 285  
   vtss\_vcap\_u48\_t, 286  
   vtss\_vcap\_u8\_t, 287  
   vtss\_vcap\_vid\_t, 289  
   vtss\_vcap\_vr\_t, 291  
 vga\_adc\_debug  
   vtss\_phy\_api.h, 598  
 vid  
   vtss\_ace\_vlan\_t, 50  
   vtss\_ece\_tag\_t, 79  
   vtss\_evc\_inner\_tag\_t, 87  
   vtss\_evc\_pb\_conf\_t, 89  
   vtss\_mce\_action\_t, 117  
   vtss\_mce\_key\_t, 118  
   vtss\_packet\_frame\_info\_t, 125  
   vtss\_packet\_port\_info\_t, 128  
   vtss\_qce\_tag\_t, 245  
   vtss\_tci\_t, 270  
   vtss\_vce\_action\_t, 292  
   vtss\_vce\_tag\_t, 304  
   vtss\_vid\_mac\_t, 306  
   vtss\_vlan\_tag\_t, 310  
   vtss\_vlan\_trans\_grp2vlan\_conf\_t, 311  
 vid\_mac  
   vtss\_mac\_table\_entry\_t, 113  
 vid\_mode  
   vtss\_evc\_inner\_tag\_t, 87  
 wlan  
   vtss\_ace\_t, 47  
   vtss\_routing\_entry\_t, 260  
 vml\_format  
   vtss\_debug\_info\_t, 64  
 vr

vtss\_vcap\_vr\_t, 292  
 vtss\_ace\_add  
   vtss\_security\_api.h, 656  
 vtss\_ace\_bit\_t  
   vtss\_security\_api.h, 652  
 vtss\_ace\_counter\_clear  
   vtss\_security\_api.h, 657  
 vtss\_ace\_counter\_get  
   vtss\_security\_api.h, 657  
 vtss\_ace\_del  
   vtss\_security\_api.h, 656  
 vtss\_ace\_frame\_arp\_t, 27  
   arp, 28  
   dip, 30  
   dmac\_match, 29  
   ethernet, 29  
   ip, 29  
   length, 29  
   req, 28  
   sip, 29  
   smac, 28  
   smac\_match, 28  
   unknown, 28  
 vtss\_ace\_frame\_etype\_t, 30  
   data, 31  
   dmac, 30  
   etype, 31  
   ptp, 31  
   smac, 31  
 vtss\_ace\_frame\_ipv4\_t, 32  
   data, 34  
   dip, 33  
   dport, 34  
   ds, 33  
   fragment, 32  
   options, 33  
   proto, 33  
   ptp, 36  
   seq\_zero, 36  
   sip, 33  
   sip\_eq\_dip, 35  
   sip\_smac, 36  
   sport, 34  
   sport\_eq\_dport, 36  
   tcp\_ack, 35  
   tcp\_fin, 34  
   tcp\_psh, 35  
   tcp\_RST, 35  
   tcp\_SYN, 34  
   tcp\_URG, 35  
   ttl, 32  
 vtss\_ace\_frame\_ipv6\_t, 37  
   data, 38  
   dport, 38  
   ds, 38  
   proto, 37  
   ptp, 40  
   seq\_zero, 40

sip, 37  
    sip\_eq\_dip, 40  
    sport, 38  
    sport\_eq\_dport, 40  
    tcp\_ack, 39  
    tcp\_fin, 39  
    tcp\_psh, 39  
    tcp\_RST, 39  
    tcp\_SYN, 39  
    tcp\_URG, 40  
    ttl, 38  
vtss\_ace\_frame\_llc\_t, 41  
    dmac, 41  
    llc, 42  
    smac, 41  
vtss\_ace\_frame\_snap\_t, 42  
    dmac, 42  
    smac, 43  
    snap, 43  
vtss\_ace\_init  
    vtss\_security\_api.h, 655  
vtss\_ace\_ptp\_t, 43  
    enable, 44  
    header, 44  
vtss\_ace\_sip\_smac\_t, 44  
    enable, 45  
    sip, 45  
    smac, 45  
vtss\_ace\_t, 45  
    action, 47  
    arp, 48  
    dmac\_bc, 47  
    dmac\_mc, 47  
    etype, 48  
    frame, 49  
    id, 46  
    ipv4, 48  
    ipv6, 49  
    llc, 48  
    policy, 46  
    port\_list, 46  
    snap, 48  
    type, 47  
    vlan, 47  
vtss\_ace\_type\_t  
    vtss\_security\_api.h, 651  
vtss\_ace\_vlan\_t, 49  
    cfi, 50  
    tagged, 50  
    usr\_prio, 50  
    vid, 50  
vtss\_acl\_action\_t, 51  
    cpu, 51  
    cpu\_once, 51  
    cpu\_queue, 51  
    evc\_police, 52  
    evc\_policer\_id, 52  
    learn, 52  
    mirror, 53  
    police, 52  
    policer\_no, 52  
    port\_action, 53  
    port\_list, 53  
    ptp\_action, 53  
vtss\_acl\_policer\_conf\_get  
    vtss\_security\_api.h, 653  
vtss\_acl\_policer\_conf\_set  
    vtss\_security\_api.h, 653  
vtss\_acl\_policer\_conf\_t, 54  
    bit\_rate, 54  
    bit\_rate\_enable, 54  
    rate, 54  
vtss\_acl\_port\_action\_t  
    vtss\_security\_api.h, 651  
vtss\_acl\_port\_conf\_get  
    vtss\_security\_api.h, 654  
vtss\_acl\_port\_conf\_set  
    vtss\_security\_api.h, 654  
vtss\_acl\_port\_conf\_t, 55  
    action, 55  
    policy\_no, 55  
vtss\_acl\_port\_counter\_clear  
    vtss\_security\_api.h, 655  
vtss\_acl\_port\_counter\_get  
    vtss\_security\_api.h, 655  
vtss\_acl\_ptp\_action\_t  
    vtss\_security\_api.h, 651  
vtss\_aggr\_mode\_get  
    vtss\_l2\_api.h, 451  
vtss\_aggr\_mode\_set  
    vtss\_l2\_api.h, 452  
vtss\_aggr\_mode\_t, 56  
    dmac\_enable, 56  
    sip\_dip\_enable, 57  
    smac\_enable, 56  
    sport\_dport\_enable, 57  
vtss\_aggr\_port\_members\_get  
    vtss\_l2\_api.h, 450  
vtss\_aggr\_port\_members\_set  
    vtss\_l2\_api.h, 451  
vtss\_aneg\_t, 57  
    generate\_pause, 58  
    obey\_pause, 58  
vtss\_api/include/vtss/api/l2\_types.h, 315  
vtss\_api/include/vtss/api/options.h, 316  
vtss\_api/include/vtss/api/phy.h, 332  
vtss\_api/include/vtss/api/port.h, 334  
vtss\_api/include/vtss/api/types.h, 347  
vtss\_api/include/vtss\_ae\_api.h, 384  
vtss\_api/include/vtss\_afi\_api.h, 384  
vtss\_api/include/vtss\_aneg\_api.h, 384  
vtss\_api/include/vtss\_api.h, 385  
vtss\_api/include/vtss\_evc\_api.h, 385  
vtss\_api/include/vtss\_fdma\_api.h, 395  
vtss\_api/include/vtss\_gfp\_api.h, 396  
vtss\_api/include/vtss\_hqos\_api.h, 396

vtss\_api/include/vtss\_i2c\_api.h, 396  
vtss\_api/include/vtss\_init\_api.h, 397  
vtss\_api/include/vtss\_l2\_api.h, 410  
vtss\_api/include/vtss\_l3\_api.h, 468  
vtss\_api/include/vtss\_mac10g\_api.h, 468  
vtss\_api/include/vtss\_misc\_api.h, 468  
vtss\_api/include/vtss\_mpls\_api.h, 503  
vtss\_api/include/vtss\_oam\_api.h, 503  
vtss\_api/include/vtss\_oha\_api.h, 503  
vtss\_api/include/vtss\_os.h, 504  
vtss\_api/include/vtss\_os\_custom.h, 504  
vtss\_api/include/vtss\_os\_ecos.h, 509  
vtss\_api/include/vtss\_os\_linux.h, 519  
vtss\_api/include/vtss\_otn\_api.h, 526  
vtss\_api/include/vtss\_packet\_api.h, 526  
vtss\_api/include/vtss\_pcs\_10gbase\_r\_api.h, 548  
vtss\_api/include/vtss\_phy\_10g\_api.h, 548  
vtss\_api/include/vtss\_phy\_api.h, 548  
vtss\_api/include/vtss\_phy\_ts\_api.h, 625  
vtss\_api/include/vtss\_port\_api.h, 626  
vtss\_api/include/vtss\_qos\_api.h, 638  
vtss\_api/include/vtss\_rab\_api.h, 647  
vtss\_api/include/vtss\_security\_api.h, 647  
vtss\_api/include/vtss\_sf14\_api.h, 657  
vtss\_api/include/vtss\_sync\_api.h, 658  
vtss\_api/include/vtss\_tfi5\_api.h, 661  
vtss\_api/include/vtss\_ts\_api.h, 662  
vtss\_api/include/vtss\_upi\_api.h, 678  
vtss\_api/include/vtss\_wis\_api.h, 678  
vtss\_api/include/vtss\_xaui\_api.h, 678  
vtss\_api/include/vtss\_xfi\_api.h, 679  
vtss\_api\_lock\_t, 58  
    file, 59  
    function, 59  
    inst, 59  
    line, 59  
vtss\_apvlan\_port\_members\_get  
    vtss\_l2\_api.h, 447  
vtss\_apvlan\_port\_members\_set  
    vtss\_l2\_api.h, 448  
vtss\_auth\_port\_state\_get  
    vtss\_security\_api.h, 652  
vtss\_auth\_port\_state\_set  
    vtss\_security\_api.h, 652  
vtss\_auth\_state\_t  
    vtss\_security\_api.h, 650  
vtss\_basic\_counters\_t, 60  
    rx\_frames, 60  
    tx\_frames, 60  
vtss\_callout\_free  
    vtss\_os\_ecos.h, 518  
vtss\_callout\_lock  
    vtss\_misc\_api.h, 484  
vtss\_callout\_malloc  
    vtss\_os\_ecos.h, 518  
vtss\_callout\_trace\_hex\_dump  
    vtss\_misc\_api.h, 482  
vtss\_callout\_trace\_printf  
    vtss\_misc\_api.h, 482  
vtss\_callout\_unlock  
    vtss\_misc\_api.h, 484  
vtss\_chip\_id\_get  
    vtss\_misc\_api.h, 487  
vtss\_chip\_id\_t, 60  
    part\_number, 61  
    revision, 61  
vtss\_counter\_pair\_t, 61  
    bytes, 62  
    frames, 62  
vtss\_debug\_group\_t  
    vtss\_misc\_api.h, 478  
vtss\_debug\_info\_get  
    vtss\_misc\_api.h, 483  
vtss\_debug\_info\_print  
    vtss\_misc\_api.h, 483  
vtss\_debug\_info\_t, 62  
    chip\_no, 63  
    clear, 64  
    full, 63  
    group, 63  
    layer, 63  
    port\_list, 63  
    vml\_format, 64  
vtss\_debug\_layer\_t  
    vtss\_misc\_api.h, 476  
vtss\_debug\_lock  
    vtss\_misc\_api.h, 484  
vtss\_debug\_lock\_t, 64  
    chip\_no, 65  
vtss\_debug\_reg\_check\_set  
    vtss\_misc\_api.h, 502  
vtss\_debug\_unlock  
    vtss\_misc\_api.h, 485  
vtss\_dev\_all\_event\_enable  
    vtss\_misc\_api.h, 489  
vtss\_dev\_all\_event\_poll  
    vtss\_misc\_api.h, 488  
vtss\_dgroup\_port\_conf\_get  
    vtss\_l2\_api.h, 448  
vtss\_dgroup\_port\_conf\_set  
    vtss\_l2\_api.h, 448  
vtss\_dgroup\_port\_conf\_t, 65  
    dgroup\_no, 65  
vtss\_dlb\_policer\_conf\_t, 66  
    cbs, 67  
    cf, 66  
    cir, 67  
    ebs, 67  
    eir, 67  
    enable, 66  
    line\_rate, 67  
    type, 66  
vtss\_dscp\_emode\_t  
    vtss\_qos\_api.h, 642  
vtss\_dscp\_mode\_t  
    vtss\_qos\_api.h, 642

vtss\_ece\_action\_t, 68  
dir, 68  
evc\_id, 69  
outer\_tag, 69  
policy\_no, 69  
pop\_tag, 68  
prio, 69  
prio\_enable, 69  
vtss\_ece\_add  
  vtss\_evc\_api.h, 393  
vtss\_ece\_del  
  vtss\_evc\_api.h, 394  
vtss\_ece\_dir\_t  
  types.h, 382  
vtss\_ece\_frame\_ipv4\_t, 70  
  dport, 71  
  dscp, 70  
  fragment, 70  
  proto, 71  
  sip, 71  
  sport, 71  
vtss\_ece\_frame\_ipv6\_t, 72  
  dport, 73  
  dscp, 72  
  proto, 72  
  sip, 72  
  sport, 73  
vtss\_ece\_init  
  vtss\_evc\_api.h, 393  
vtss\_ece\_key\_t, 73  
  frame, 75  
  ipv4, 74  
  ipv6, 75  
  mac, 74  
  port\_list, 74  
  tag, 74  
  type, 74  
vtss\_ece\_mac\_t, 75  
  dmac\_bc, 76  
  dmac\_mc, 76  
  smac, 76  
vtss\_ece\_outer\_tag\_t, 76  
  dei, 77  
  enable, 77  
  pcp, 77  
  pcp\_dei\_preserve, 77  
vtss\_ece\_pop\_tag\_t  
  types.h, 383  
vtss\_ece\_port\_t  
  vtss\_evc\_api.h, 389  
vtss\_ece\_t, 78  
  action, 78  
  id, 78  
  key, 78  
vtss\_ece\_tag\_t, 79  
  dei, 80  
  pcp, 79  
  s\_tagged, 80  
             tagged, 80  
             vid, 79  
vtss\_ece\_type\_t  
  vtss\_evc\_api.h, 389  
vtss\_eee\_port\_conf\_set  
  vtss\_misc\_api.h, 501  
vtss\_eee\_port\_conf\_t, 80  
  eee\_ena, 81  
  eee\_fast\_queues, 81  
  lp\_advertisement, 81  
  optimized\_for\_power, 82  
  tx\_tw, 81  
vtss\_eee\_port\_counter\_get  
  vtss\_misc\_api.h, 502  
vtss\_eee\_port\_counter\_t, 82  
  fill\_level, 83  
  fill\_level\_get, 82  
  fill\_level\_thres, 83  
  tx\_out\_bytes, 83  
  tx\_out\_bytes\_get, 83  
vtss\_eee\_port\_state\_set  
  vtss\_misc\_api.h, 501  
vtss\_eee\_port\_state\_t, 84  
  select, 84  
  val, 84  
vtss\_eee\_state\_select\_t  
  vtss\_misc\_api.h, 480  
vtss\_eps\_port\_conf\_get  
  vtss\_l2\_api.h, 463  
vtss\_eps\_port\_conf\_set  
  vtss\_l2\_api.h, 464  
vtss\_eps\_port\_conf\_t, 84  
  port\_no, 85  
  type, 85  
vtss\_eps\_port\_selector\_get  
  vtss\_l2\_api.h, 464  
vtss\_eps\_port\_selector\_set  
  vtss\_l2\_api.h, 464  
vtss\_eps\_port\_type\_t  
  vtss\_l2\_api.h, 426  
vtss\_eps\_selector\_t  
  vtss\_l2\_api.h, 426  
vtss\_erps\_port\_state\_get  
  vtss\_l2\_api.h, 467  
vtss\_erps\_port\_state\_set  
  vtss\_l2\_api.h, 467  
vtss\_erps\_state\_t  
  vtss\_l2\_api.h, 426  
vtss\_erps\_vlan\_member\_get  
  vtss\_l2\_api.h, 465  
vtss\_erps\_vlan\_member\_set  
  vtss\_l2\_api.h, 465  
vtss\_evc\_add  
  vtss\_evc\_api.h, 391  
vtss\_evc\_api.h  
  VTSS\_ECE\_ID\_LAST, 388  
  VTSS\_EVC\_ID\_NONE, 387  
  VTSS\_EVC\_POLICERS, 387

VTSS\_MCE\_ID\_LAST, 388  
 VTSS\_MCE\_POP\_NONE, 388  
 vtss\_ece\_add, 393  
 vtss\_ece\_del, 394  
 vtss\_ece\_init, 393  
 vtss\_ece\_port\_t, 389  
 vtss\_ece\_type\_t, 389  
 vtss\_evc\_add, 391  
 vtss\_evc\_del, 392  
 vtss\_evc\_get, 392  
 vtss\_evc\_inner\_tag\_type\_t, 389  
 vtss\_evc\_policer\_conf\_get, 391  
 vtss\_evc\_policer\_conf\_set, 391  
 vtss\_evc\_port\_conf\_get, 390  
 vtss\_evc\_port\_conf\_set, 390  
 vtss\_evc\_vid\_mode\_t, 388  
 vtss\_mce\_add, 394  
 vtss\_mce\_del, 395  
 vtss\_mce\_init, 394  
 vtss\_evc\_conf\_t, 85  
     learning, 86  
     network, 86  
     pb, 86  
 vtss\_evc\_del  
     vtss\_evc\_api.h, 392  
 vtss\_evc\_get  
     vtss\_evc\_api.h, 392  
 vtss\_evc\_inner\_tag\_t, 87  
     dei, 88  
     pcp, 88  
     pcp\_dei\_preserve, 88  
     type, 87  
     vid, 87  
     vid\_mode, 87  
 vtss\_evc\_inner\_tag\_type\_t  
     vtss\_evc\_api.h, 389  
 vtss\_evc\_pb\_conf\_t, 88  
     inner\_tag, 90  
     ivid, 89  
     nni, 89  
     uvvid, 89  
     vid, 89  
 vtss\_evc\_policer\_conf\_get  
     vtss\_evc\_api.h, 391  
 vtss\_evc\_policer\_conf\_set  
     vtss\_evc\_api.h, 391  
 vtss\_evc\_port\_conf\_get  
     vtss\_evc\_api.h, 390  
 vtss\_evc\_port\_conf\_set  
     vtss\_evc\_api.h, 390  
 vtss\_evc\_port\_conf\_t, 90  
     dei\_colouring, 90  
     dmac\_dip, 91  
     inner\_tag, 91  
 vtss\_evc\_vid\_mode\_t  
     vtss\_evc\_api.h, 388  
 vtss\_fan\_conf\_t, 91  
     fan\_low\_pol, 92  
         fan\_open\_col, 92  
         fan\_pwm\_freq, 92  
         ppr, 92  
         type, 92  
 vtss\_fan\_controller\_init  
     vtss\_misc\_api.h, 500  
 vtss\_fan\_cool\_lvl\_get  
     vtss\_misc\_api.h, 500  
 vtss\_fan\_cool\_lvl\_set  
     vtss\_misc\_api.h, 500  
 vtss\_fan\_rotation\_get  
     vtss\_misc\_api.h, 499  
 vtss\_fefi\_mode\_t  
     vtss\_phy\_api.h, 579  
 vtss\_fiber\_port\_speed\_t  
     port.h, 346  
 vtss\_gpio\_direction\_set  
     vtss\_misc\_api.h, 490  
 vtss\_gpio\_event\_enable  
     vtss\_misc\_api.h, 492  
 vtss\_gpio\_event\_poll  
     vtss\_misc\_api.h, 491  
 vtss\_gpio\_mode\_set  
     vtss\_misc\_api.h, 489  
 vtss\_gpio\_mode\_t  
     vtss\_misc\_api.h, 479  
 vtss\_gpio\_read  
     vtss\_misc\_api.h, 490  
 vtss\_gpio\_write  
     vtss\_misc\_api.h, 491  
 vtss\_hqos\_sch\_mode\_t  
     types.h, 383  
 vtss\_i2c\_read\_t  
     vtss\_init\_api.h, 400  
 vtss\_i2c\_write\_t  
     vtss\_init\_api.h, 401  
 vtss\_init\_api.h  
     VTSS\_I2C\_NO\_MULTIPLEXER, 399  
     vtss\_i2c\_read\_t, 400  
     vtss\_i2c\_write\_t, 401  
     vtss\_init\_conf\_get, 408  
     vtss\_init\_conf\_set, 408  
     vtss\_inst\_create, 407  
     vtss\_inst\_destroy, 408  
     vtss\_inst\_get, 407  
     vtss\_miim\_read\_t, 403  
     vtss\_miim\_write\_t, 403  
     vtss\_mmd\_read\_inc\_t, 404  
     vtss\_mmd\_read\_t, 403  
     vtss\_mmd\_write\_t, 404  
     vtss\_port\_mux\_mode\_t, 406  
     vtss\_reg\_read\_t, 399  
     vtss\_reg\_write\_t, 400  
     vtss\_restart\_conf\_end, 409  
     vtss\_restart\_conf\_get, 409  
     vtss\_restart\_conf\_set, 410  
     vtss\_restart\_status\_get, 409  
     vtss\_restart\_t, 406

vtss\_spi\_32bit\_read\_write\_t, 402  
vtss\_spi\_64bit\_read\_write\_t, 402  
vtss\_spi\_read\_write\_t, 401  
vtss\_target\_type\_t, 405  
vtss\_init\_conf\_get  
    vtss\_init\_api.h, 408  
vtss\_init\_conf\_set  
    vtss\_init\_api.h, 408  
vtss\_init\_conf\_t, 93  
    miim\_read, 94  
    miim\_write, 94  
    mmd\_read, 94  
    mmd\_read\_inc, 94  
    mmd\_write, 95  
    mux\_mode, 96  
    pi, 96  
    reg\_read, 93  
    reg\_write, 94  
    restart\_info\_port, 96  
    restart\_info\_src, 96  
    serdes, 96  
    spi\_32bit\_read\_write, 95  
    spi\_64bit\_read\_write, 95  
    spi\_read\_write, 95  
    warm\_start\_enable, 95  
vtss\_inst\_create  
    vtss\_init\_api.h, 407  
vtss\_inst\_create\_t, 97  
    target, 97  
vtss\_inst\_destroy  
    vtss\_init\_api.h, 408  
vtss\_inst\_get  
    vtss\_init\_api.h, 407  
vtss\_internal\_bw\_t  
    vtss\_port\_api.h, 629  
vtss\_intr\_cfg  
    vtss\_misc\_api.h, 494  
vtss\_intr\_mask\_set  
    vtss\_misc\_api.h, 495  
vtss\_intr\_pol\_negation  
    vtss\_misc\_api.h, 496  
vtss\_intr\_status\_get  
    vtss\_misc\_api.h, 495  
vtss\_intr\_sticky\_clear  
    vtss\_misc\_api.h, 486  
vtss\_intr\_t, 98  
    link\_change, 98  
vtss\_ip\_addr\_t, 98  
    addr, 99  
    ipv4, 99  
    ipv6, 99  
    type, 99  
vtss\_ip\_network\_t, 100  
    address, 100  
    prefix\_size, 100  
vtss\_ip\_type\_t  
    types.h, 381  
vtss\_ipv4\_mc\_add  
    vtss\_l2\_api.h, 459  
vtss\_ipv4\_mc\_del  
    vtss\_l2\_api.h, 459  
vtss\_ipv4\_mc\_flood\_members\_get  
    vtss\_l2\_api.h, 458  
vtss\_ipv4\_mc\_flood\_members\_set  
    vtss\_l2\_api.h, 458  
vtss\_ipv4\_network\_t, 100  
    address, 101  
    prefix\_size, 101  
vtss\_ipv4\_uc\_t, 101  
    destination, 102  
    network, 102  
vtss\_ipv6\_mc\_add  
    vtss\_l2\_api.h, 462  
vtss\_ipv6\_mc\_ctrl\_flood\_get  
    vtss\_l2\_api.h, 462  
vtss\_ipv6\_mc\_ctrl\_flood\_set  
    vtss\_l2\_api.h, 462  
vtss\_ipv6\_mc\_del  
    vtss\_l2\_api.h, 463  
vtss\_ipv6\_mc\_flood\_members\_get  
    vtss\_l2\_api.h, 461  
vtss\_ipv6\_mc\_flood\_members\_set  
    vtss\_l2\_api.h, 461  
vtss\_ipv6\_network\_t, 102  
    address, 103  
    prefix\_size, 103  
vtss\_ipv6\_t, 103  
    addr, 104  
vtss\_ipv6\_uc\_t, 104  
    destination, 104  
    network, 104  
vtss\_irq\_conf\_get  
    vtss\_misc\_api.h, 496  
vtss\_irq\_conf\_set  
    vtss\_misc\_api.h, 496  
vtss\_irq\_conf\_t, 105  
    destination, 105  
    external, 105  
vtss\_irq\_enable  
    vtss\_misc\_api.h, 497  
vtss\_irq\_status\_get\_and\_mask  
    vtss\_misc\_api.h, 497  
vtss\_irq\_status\_t, 106  
    active, 106  
    raw\_ident, 106  
    raw\_mask, 107  
    raw\_status, 107  
vtss\_irq\_t  
    vtss\_misc\_api.h, 480  
vtss\_isdx\_t  
    types.h, 375  
vtss\_isolated\_port\_members\_get  
    vtss\_l2\_api.h, 445  
vtss\_isolated\_port\_members\_set  
    vtss\_l2\_api.h, 446  
vtss\_isolated\_vlan\_get

vtss\_l2\_api.h, 445  
 vtss\_isolated\_vlan\_set  
   vtss\_l2\_api.h, 445  
 vtss\_l2\_api.h  
   VTSS\_ERPI\_ARRAY\_SIZE, 424  
   VTSS\_ERPI\_END, 424  
   VTSS\_ERPI\_START, 423  
   VTSS\_ERPIS, 423  
   VTSS\_MAC\_ADDRS, 418  
   VTSS\_MSTI\_ARRAY\_SIZE, 418  
   VTSS\_MSTI\_END, 418  
   VTSS\_MSTI\_START, 418  
   VTSS\_MSTIS, 418  
   VTSS\_PVLAN\_ARRAY\_SIZE, 423  
   VTSS\_PVLAN\_NO\_DEFAULT, 423  
   VTSS\_PVLAN\_NO\_END, 423  
   VTSS\_PVLAN\_NO\_START, 422  
   VTSS\_PVLANS, 422  
   VTSS\_VCE\_ID\_LAST, 419  
   VTSS\_VCL\_ARRAY\_SIZE, 419  
   VTSS\_VCL\_ID\_END, 419  
   VTSS\_VCL\_ID\_START, 419  
   VTSS\_VCL\_IDS, 419  
   VTSS\_VLAN\_TRANS\_FIRST\_GROUP\_ID, 420  
   VTSS\_VLAN\_TRANS\_GROUP\_MAX\_CNT, 420  
   VTSS\_VLAN\_TRANS\_LAST\_GROUP\_ID, 421  
   VTSS\_VLAN\_TRANS\_MAX\_CNT, 420  
   VTSS\_VLAN\_TRANS\_MAX\_VLAN\_ID, 421  
   VTSS\_VLAN\_TRANS\_NULL\_CHECK, 422  
   VTSS\_VLAN\_TRANS\_NULL\_GROUP\_ID, 420  
   VTSS\_VLAN\_TRANS\_PORT\_BF\_SIZE, 422  
   VTSS\_VLAN\_TRANS\_VALID\_GROUP\_CHECK,  
     421  
   VTSS\_VLAN\_TRANS\_VALID\_VLAN\_CHECK,  
     421  
   VTSS\_VLAN\_TRANS\_VID\_START, 420  
 vtss\_aggr\_mode\_get, 451  
 vtss\_aggr\_mode\_set, 452  
 vtss\_aggr\_port\_members\_get, 450  
 vtss\_aggr\_port\_members\_set, 451  
 vtss\_apvlan\_port\_members\_get, 447  
 vtss\_apvlan\_port\_members\_set, 448  
 vtss\_dgroup\_port\_conf\_get, 448  
 vtss\_dgroup\_port\_conf\_set, 448  
 vtss\_eps\_port\_conf\_get, 463  
 vtss\_eps\_port\_conf\_set, 464  
 vtss\_eps\_port\_selector\_get, 464  
 vtss\_eps\_port\_selector\_set, 464  
 vtss\_eps\_port\_type\_t, 426  
 vtss\_eps\_selector\_t, 426  
 vtss\_erps\_port\_state\_get, 467  
 vtss\_erps\_port\_state\_set, 467  
 vtss\_erps\_state\_t, 426  
 vtss\_erps\_vlan\_member\_get, 465  
 vtss\_erps\_vlan\_member\_set, 465  
 vtss\_ipv4\_mc\_add, 459  
 vtss\_ipv4\_mc\_del, 459  
 vtss\_ipv4\_mc\_flood\_members\_get, 458  
 vtss\_ipv4\_mc\_flood\_members\_set, 458  
 vtss\_ipv6\_mc\_add, 462  
 vtss\_ipv6\_mc\_ctrl\_flood\_get, 462  
 vtss\_ipv6\_mc\_ctrl\_flood\_set, 462  
 vtss\_ipv6\_mc\_del, 463  
 vtss\_ipv6\_mc\_flood\_members\_get, 461  
 vtss\_ipv6\_mc\_flood\_members\_set, 461  
 vtss\_isolated\_port\_members\_get, 445  
 vtss\_isolated\_port\_members\_set, 446  
 vtss\_isolated\_vlan\_get, 445  
 vtss\_isolated\_vlan\_set, 445  
 vtss\_learn\_port\_mode\_get, 432  
 vtss\_learn\_port\_mode\_set, 432  
 vtss\_mac\_table\_add, 427  
 vtss\_mac\_table\_age, 429  
 vtss\_mac\_table\_age\_time\_get, 428  
 vtss\_mac\_table\_age\_time\_set, 429  
 vtss\_mac\_table\_del, 427  
 vtss\_mac\_table\_flush, 430  
 vtss\_mac\_table\_get, 427  
 vtss\_mac\_table\_get\_next, 428  
 vtss\_mac\_table\_port\_flush, 430  
 vtss\_mac\_table\_status\_get, 431  
 vtss\_mac\_table\_vlan\_age, 429  
 vtss\_mac\_table\_vlan\_flush, 431  
 vtss\_mac\_table\_vlan\_port\_flush, 431  
 vtss\_mc\_flood\_members\_get, 457  
 vtss\_mc\_flood\_members\_set, 458  
 vtss\_mirror\_conf\_get, 452  
 vtss\_mirror\_conf\_set, 452  
 vtss\_mirror\_cpu\_egress\_get, 456  
 vtss\_mirror\_cpu\_egress\_set, 456  
 vtss\_mirror\_cpu\_ingress\_get, 455  
 vtss\_mirror\_cpu\_ingress\_set, 455  
 vtss\_mirror\_egress\_ports\_get, 454  
 vtss\_mirror\_egress\_ports\_set, 455  
 vtss\_mirror\_ingress\_ports\_get, 453  
 vtss\_mirror\_ingress\_ports\_set, 454  
 vtss\_mirror\_monitor\_port\_get, 453  
 vtss\_mirror\_monitor\_port\_set, 453  
 vtss\_mstp\_port\_msti\_state\_get, 435  
 vtss\_mstp\_port\_msti\_state\_set, 436  
 vtss\_mstp\_vlan\_msti\_get, 434  
 vtss\_mstp\_vlan\_msti\_set, 435  
 vtss\_port\_state\_get, 433  
 vtss\_port\_state\_set, 433  
 vtss\_pvlan\_port\_members\_get, 446  
 vtss\_pvlan\_port\_members\_set, 447  
 vtss\_sfflow\_port\_conf\_get, 449  
 vtss\_sfflow\_port\_conf\_set, 449  
 vtss\_sfflow\_sampling\_rate\_convert, 450  
 vtss\_stp\_port\_state\_get, 433  
 vtss\_stp\_port\_state\_set, 434  
 vtss\_stp\_state\_t, 424  
 vtss\_uc\_flood\_members\_get, 457  
 vtss\_uc\_flood\_members\_set, 457  
 vtss\_vce\_add, 442  
 vtss\_vce\_del, 442

vtss\_vce\_init, 441  
vtss\_vce\_type\_t, 425  
vtss\_vcl\_port\_conf\_get, 440  
vtss\_vcl\_port\_conf\_set, 441  
vtss\_vlan\_conf\_get, 436  
vtss\_vlan\_conf\_set, 436  
vtss\_vlan\_port\_conf\_get, 437  
vtss\_vlan\_port\_conf\_set, 437  
vtss\_vlan\_port\_members\_get, 438  
vtss\_vlan\_port\_members\_set, 438  
vtss\_vlan\_port\_type\_t, 425  
vtss\_vlan\_trans\_group\_add, 442  
vtss\_vlan\_trans\_group\_del, 443  
vtss\_vlan\_trans\_group\_get, 443  
vtss\_vlan\_trans\_group\_to\_port\_get, 444  
vtss\_vlan\_trans\_group\_to\_port\_set, 444  
vtss\_vlan\_tx\_tag\_get, 439  
vtss\_vlan\_tx\_tag\_set, 440  
vtss\_vlan\_tx\_tag\_t, 425  
vtss\_vlan\_vid\_conf\_get, 439  
vtss\_vlan\_vid\_conf\_set, 439  
vtss\_vt\_id\_t, 424  
vtss\_l3\_counters\_t, 107  
  ipv4uc\_received\_frames, 108  
  ipv4uc\_received\_octets, 108  
  ipv4uc\_transmitted\_frames, 109  
  ipv4uc\_transmitted\_octets, 108  
  ipv6uc\_received\_frames, 108  
  ipv6uc\_received\_octets, 108  
  ipv6uc\_transmitted\_frames, 109  
  ipv6uc\_transmitted\_octets, 109  
vtss\_lcpll\_status\_t, 109  
  cal\_done, 110  
  cal\_error, 110  
  fsm\_lock, 110  
  fsm\_stat, 111  
  gain\_stat, 111  
  lock\_status, 110  
vtss\_learn\_mode\_t, 111  
  automatic, 112  
  cpu, 112  
  discard, 112  
vtss\_learn\_port\_mode\_get  
  vtss\_l2\_api.h, 432  
vtss\_learn\_port\_mode\_set  
  vtss\_l2\_api.h, 432  
vtss\_mac\_addr\_t  
  types.h, 375  
vtss\_mac\_t, 112  
  addr, 113  
vtss\_mac\_table\_add  
  vtss\_l2\_api.h, 427  
vtss\_mac\_table\_age  
  vtss\_l2\_api.h, 429  
vtss\_mac\_table\_age\_time\_get  
  vtss\_l2\_api.h, 428  
vtss\_mac\_table\_age\_time\_set  
  vtss\_l2\_api.h, 429  
vtss\_mac\_table\_del  
  vtss\_l2\_api.h, 427  
vtss\_mac\_table\_entry\_t, 113  
  aged, 114  
  copy\_to\_cpu, 114  
  cpu\_queue, 114  
  destination, 114  
  locked, 114  
  vid\_mac, 113  
vtss\_mac\_table\_flush  
  vtss\_l2\_api.h, 430  
vtss\_mac\_table\_get  
  vtss\_l2\_api.h, 427  
vtss\_mac\_table\_get\_next  
  vtss\_l2\_api.h, 428  
vtss\_mac\_table\_port\_flush  
  vtss\_l2\_api.h, 430  
vtss\_mac\_table\_status\_get  
  vtss\_l2\_api.h, 431  
vtss\_mac\_table\_status\_t, 115  
  aged, 116  
  learned, 115  
  moved, 116  
  replaced, 115  
vtss\_mac\_table\_vlan\_age  
  vtss\_l2\_api.h, 429  
vtss\_mac\_table\_vlan\_flush  
  vtss\_l2\_api.h, 431  
vtss\_mac\_table\_vlan\_port\_flush  
  vtss\_l2\_api.h, 431  
vtss\_mc\_flood\_members\_get  
  vtss\_l2\_api.h, 457  
vtss\_mc\_flood\_members\_set  
  vtss\_l2\_api.h, 458  
vtss\_mce\_action\_t, 116  
  policy\_no, 117  
  pop\_cnt, 117  
  prio, 117  
  prio\_enable, 117  
  vid, 117  
vtss\_mce\_add  
  vtss\_evc\_api.h, 394  
vtss\_mce\_del  
  vtss\_evc\_api.h, 395  
vtss\_mce\_init  
  vtss\_evc\_api.h, 394  
vtss\_mce\_key\_t, 118  
  data, 119  
  port\_list, 118  
  vid, 118  
vtss\_mce\_t, 119  
  action, 120  
  id, 120  
  key, 120  
  tt\_loop, 119  
vtss\_mem\_flags\_t  
  types.h, 378  
vtss\_mep\_policer\_conf\_get

vtss\_qos\_api.h, 644  
 vtss\_mep\_policer\_conf\_set  
     vtss\_qos\_api.h, 644  
 vtss\_miim\_controller\_t  
     vtss\_port\_api.h, 629  
 vtss\_miim\_read  
     vtss\_port\_api.h, 637  
 vtss\_miim\_read\_t  
     vtss\_init\_api.h, 403  
 vtss\_miim\_write  
     vtss\_port\_api.h, 637  
 vtss\_miim\_write\_t  
     vtss\_init\_api.h, 403  
 vtss\_mirror\_conf\_get  
     vtss\_l2\_api.h, 452  
 vtss\_mirror\_conf\_set  
     vtss\_l2\_api.h, 452  
 vtss\_mirror\_conf\_t, 120  
     fwd\_enable, 121  
     port\_no, 121  
 vtss\_mirror\_cpu\_egress\_get  
     vtss\_l2\_api.h, 456  
 vtss\_mirror\_cpu\_egress\_set  
     vtss\_l2\_api.h, 456  
 vtss\_mirror\_cpu\_ingress\_get  
     vtss\_l2\_api.h, 455  
 vtss\_mirror\_cpu\_ingress\_set  
     vtss\_l2\_api.h, 455  
 vtss\_mirror\_egress\_ports\_get  
     vtss\_l2\_api.h, 454  
 vtss\_mirror\_egress\_ports\_set  
     vtss\_l2\_api.h, 455  
 vtss\_mirror\_ingress\_ports\_get  
     vtss\_l2\_api.h, 453  
 vtss\_mirror\_ingress\_ports\_set  
     vtss\_l2\_api.h, 454  
 vtss\_mirror\_monitor\_port\_get  
     vtss\_l2\_api.h, 453  
 vtss\_mirror\_monitor\_port\_set  
     vtss\_l2\_api.h, 453  
 vtss\_misc\_api.h  
     tod\_get\_ns\_cnt\_cb\_t, 475  
     VTSS\_OS\_TIMESTAMP\_TYPE, 474  
     VTSS\_OS\_TIMESTAMP, 474  
     vtss\_callout\_lock, 484  
     vtss\_callout\_trace\_hex\_dump, 482  
     vtss\_callout\_trace\_printf, 482  
     vtss\_callout\_unlock, 484  
     vtss\_chip\_id\_get, 487  
     vtss\_debug\_group\_t, 478  
     vtss\_debug\_info\_get, 483  
     vtss\_debug\_info\_print, 483  
     vtss\_debug\_layer\_t, 476  
     vtss\_debug\_lock, 484  
     vtss\_debug\_reg\_check\_set, 502  
     vtss\_debug\_unlock, 485  
     vtss\_dev\_all\_event\_enable, 489  
     vtss\_dev\_all\_event\_poll, 488  
     vtss\_eee\_port\_conf\_set, 501  
     vtss\_eee\_port\_counter\_get, 502  
     vtss\_eee\_port\_state\_set, 501  
     vtss\_eee\_state\_select\_t, 480  
     vtss\_fan\_controller\_init, 500  
     vtss\_fan\_cool\_lvl\_get, 500  
     vtss\_fan\_cool\_lvl\_set, 500  
     vtss\_fan\_rotation\_get, 499  
     vtss\_gpio\_direction\_set, 490  
     vtss\_gpio\_event\_enable, 492  
     vtss\_gpio\_event\_poll, 491  
     vtss\_gpio\_mode\_set, 489  
     vtss\_gpio\_mode\_t, 479  
     vtss\_gpio\_read, 490  
     vtss\_gpio\_write, 491  
     vtss\_intr\_cfg, 494  
     vtss\_intr\_mask\_set, 495  
     vtss\_intr\_pol\_negation, 496  
     vtss\_intr\_status\_get, 495  
     vtss\_intr\_sticky\_clear, 486  
     vtss\_irq\_conf\_get, 496  
     vtss\_irq\_conf\_set, 496  
     vtss\_irq\_enable, 497  
     vtss\_irq\_status\_get\_and\_mask, 497  
     vtss\_irq\_t, 480  
     vtss\_poll\_1sec, 487  
     vtss\_ptp\_event\_enable, 488  
     vtss\_ptp\_event\_poll, 487  
     vtss\_reg\_read, 485  
     vtss\_reg\_write, 485  
     vtss\_reg\_write\_masked, 486  
     vtss\_sgpio\_bmode\_t, 480  
     vtss\_sgpio\_conf\_get, 492  
     vtss\_sgpio\_conf\_set, 492  
     vtss\_sgpio\_event\_enable, 494  
     vtss\_sgpio\_event\_poll, 493  
     vtss\_sgpio\_mode\_t, 479  
     vtss\_sgpio\_read, 493  
     vtss\_temp\_sensor\_get, 499  
     vtss\_temp\_sensor\_init, 498  
     vtss\_tod\_get\_ns\_cnt, 498  
     vtss\_tod\_set\_ns\_cnt\_cb, 498  
     vtss\_trace\_conf\_get, 481  
     vtss\_trace\_conf\_set, 481  
     vtss\_trace\_group\_t, 475  
     vtss\_trace\_layer\_t, 475  
     vtss\_trace\_level\_t, 476  
 vtss\_mmd\_read\_inc\_t  
     vtss\_init\_api.h, 404  
 vtss\_mmd\_read\_t  
     vtss\_init\_api.h, 403  
 vtss\_mmd\_write\_t  
     vtss\_init\_api.h, 404  
 vtss\_mstp\_port\_msti\_state\_get  
     vtss\_l2\_api.h, 435  
 vtss\_mstp\_port\_msti\_state\_set  
     vtss\_l2\_api.h, 436  
 vtss\_mstp\_vlan\_msti\_get

vtss\_l2\_api.h, 434  
vtss\_mstp\_vlan\_msti\_set  
    vtss\_l2\_api.h, 435  
vtss\_mtimer\_t, 121  
    now, 122  
    timeout, 122  
    vtss\_os\_custom.h, 508  
    vtss\_os\_ecos.h, 517  
vtss\_npi\_conf\_get  
    vtss\_packet\_api.h, 535  
vtss\_npi\_conf\_set  
    vtss\_packet\_api.h, 535  
vtss\_npi\_conf\_t, 122  
    enable, 123  
    port\_no, 123  
vtss\_os\_custom.h  
    uint, 505  
    ulong, 505  
    VTSS\_DIV64, 506  
    VTSS\_LABS, 506  
    VTSS\_LLabs, 507  
    VTSS\_MOD64, 506  
    VTSS\_MSLEEP, 505  
    VTSS\_MTIMER\_CANCEL, 506  
    VTSS\_MTIMER\_START, 505  
    VTSS\_MTIMER\_TIMEOUT, 505  
    VTSS\_OS\_Ctz64, 507  
    VTSS\_OS\_Ctz, 507  
    VTSS\_OS\_Free, 508  
    VTSS\_OS\_Malloc, 507  
    VTSS\_OS\_Rand, 508  
    vtss\_mtimer\_t, 508  
vtss\_os\_ecos.h  
    llabs, 517  
    VTSS\_DIV64, 511  
    VTSS\_LABS, 512  
    VTSS\_LLabs, 512  
    VTSS\_MOD64, 511  
    VTSS\_MSLEEP, 510  
    VTSS\_MTIMER\_CANCEL, 511  
    VTSS\_MTIMER\_START, 510  
    VTSS\_MTIMER\_TIMEOUT, 511  
    VTSS\_NSLEEP, 510  
    VTSS\_OS\_BIG\_ENDIAN, 515  
    VTSS\_OS\_COMPILER\_ATTRIBUTE\_ALIGNED,  
        514  
    VTSS\_OS\_Ctz64, 512  
    VTSS\_OS\_Ctz, 512  
    VTSS\_OS\_DCACHE\_FLUSH, 515  
    VTSS\_OS\_DCACHE\_INVALIDATE, 514  
    VTSS\_OS\_DCACHE\_LINE\_SIZE\_BYTES, 514  
    VTSS\_OS\_FREE, 513  
    VTSS\_OS\_INTERRUPT\_DISABLE, 517  
    VTSS\_OS\_INTERRUPT\_FLAGS, 516  
    VTSS\_OS\_INTERRUPT\_RESTORE, 517  
    VTSS\_OS\_MALLOC, 513  
    VTSS\_OS\_NTOHL, 515  
    VTSS\_OS\_RAND, 513  
    VTSS\_OS\_reordered\_barrier, 514  
    VTSS\_OS\_SCHEDULER\_FLAGS, 516  
    VTSS\_OS\_SCHEDULER\_LOCK, 516  
    VTSS\_OS\_SCHEDULER\_UNLOCK, 516  
    VTSS\_OS\_VIRT\_TO\_PHYS, 515  
    VTSS\_TIME\_OF\_DAY, 511  
    vtss\_callout\_free, 518  
    vtss\_callout\_malloc, 518  
    vtss\_mtimer\_t, 517  
vtss\_os\_linux.h  
    VTSS\_DIV64, 523  
    VTSS\_LABS, 523  
    VTSS\_LLabs, 523  
    VTSS\_MOD64, 523  
    VTSS\_MSLEEP, 520  
    VTSS\_MTIMER\_CANCEL, 521  
    VTSS\_MTIMER\_START, 521  
    VTSS\_MTIMER\_TIMEOUT, 521  
    VTSS\_NSLEEP, 520  
    VTSS\_OS\_BIG\_ENDIAN, 520  
    VTSS\_OS\_Ctz64, 525  
    VTSS\_OS\_Ctz, 524  
    VTSS\_OS\_Free, 525  
    VTSS\_OS\_Malloc, 525  
    VTSS\_OS\_NTOHL, 520  
    VTSS\_OS\_Rand, 526  
    VTSS\_OS\_SCHEDULER\_FLAGS, 522  
    VTSS\_OS\_SCHEDULER\_LOCK, 522  
    VTSS\_OS\_SCHEDULER\_UNLOCK, 522  
    VTSS\_TIME\_OF\_DAY, 522  
vtss\_os\_timestamp\_t, 123  
    hw\_cnt, 124  
vtss\_packet\_api.h  
    VTSS\_JR1\_PACKET\_HDR\_SIZE\_BYTES, 530  
    VTSS\_JR1\_RX\_IFH\_SIZE, 531  
    VTSS\_JR2\_PACKET\_HDR\_SIZE\_BYTES, 530  
    VTSS\_JR2\_RX\_IFH\_SIZE, 531  
    VTSS\_L26\_PACKET\_HDR\_SIZE\_BYTES, 530  
    VTSS\_L26\_RX\_IFH\_SIZE, 531  
    VTSS\_PACKET\_HDR\_SIZE\_BYTES, 530  
    VTSS\_PACKET\_TX\_IFH\_MAX, 531  
    VTSS\_PRIO\_SUPER, 529  
    VTSS\_SVL\_PACKET\_HDR\_SIZE\_BYTES, 530  
    VTSS\_SVL\_RX\_IFH\_SIZE, 531  
    vtss\_npi\_conf\_get, 535  
    vtss\_npi\_conf\_set, 535  
    vtss\_packet\_dma\_conf\_get, 546  
    vtss\_packet\_dma\_conf\_set, 547  
    vtss\_packet\_dma\_offset, 547  
    vtss\_packet\_filter\_t, 532  
    vtss\_packet\_frame\_filter, 541  
    vtss\_packet\_frame\_info\_init, 541  
    vtss\_packet\_oam\_type\_t, 532  
    vtss\_packet\_port\_filter\_get, 542  
    vtss\_packet\_port\_info\_init, 542  
    vtss\_packet\_ptp\_action\_t, 533  
    vtss\_packet\_rx\_conf\_get, 536  
    vtss\_packet\_rx\_conf\_set, 536

vtss\_packet\_rx\_frame\_discard, 539  
 vtss\_packet\_rx\_frame\_get, 538  
 vtss\_packet\_rx\_frame\_get\_raw, 539  
 vtss\_packet\_rx\_hdr\_decode, 543  
 vtss\_packet\_rx\_hints\_t, 533  
 vtss\_packet\_rx\_port\_conf\_get, 536  
 vtss\_packet\_rx\_port\_conf\_set, 538  
 vtss\_packet\_tx\_frame, 545  
 vtss\_packet\_tx\_frame\_port, 540  
 vtss\_packet\_tx\_frame\_port\_vlan, 540  
 vtss\_packet\_tx\_frame\_vlan, 541  
 vtss\_packet\_tx\_hdr\_compile, 545  
 vtss\_packet\_tx\_hdr\_encode, 543  
 vtss\_packet\_tx\_info\_init, 546  
 vtss\_packet\_tx\_vstax\_t, 534  
 vtss\_tag\_type\_t, 533  
 vtss\_packet\_dma\_conf\_get  
     vtss\_packet\_api.h, 546  
 vtss\_packet\_dma\_conf\_set  
     vtss\_packet\_api.h, 547  
 vtss\_packet\_dma\_conf\_t, 124  
     dma\_enable, 124  
 vtss\_packet\_dma\_offset  
     vtss\_packet\_api.h, 547  
 vtss\_packet\_filter\_t  
     vtss\_packet\_api.h, 532  
 vtss\_packet\_frame\_filter  
     vtss\_packet\_api.h, 541  
 vtss\_packet\_frame\_info\_init  
     vtss\_packet\_api.h, 541  
 vtss\_packet\_frame\_info\_t, 125  
     aggr\_rx\_disable, 126  
     aggr\_tx\_disable, 126  
     port\_no, 125  
     port\_tx, 125  
     vid, 125  
 vtss\_packet\_oam\_type\_t  
     vtss\_packet\_api.h, 532  
 vtss\_packet\_port\_filter\_get  
     vtss\_packet\_api.h, 542  
 vtss\_packet\_port\_filter\_t, 126  
     filter, 127  
     tpid, 127  
 vtss\_packet\_port\_info\_init  
     vtss\_packet\_api.h, 542  
 vtss\_packet\_port\_info\_t, 127  
     aggr\_rx\_disable, 128  
     aggr\_tx\_disable, 128  
     port\_no, 128  
     vid, 128  
 vtss\_packet\_ptp\_action\_t  
     vtss\_packet\_api.h, 533  
 vtss\_packet\_reg\_type\_t  
     types.h, 380  
 vtss\_packet\_rx\_conf\_get  
     vtss\_packet\_api.h, 536  
 vtss\_packet\_rx\_conf\_set  
     vtss\_packet\_api.h, 536  
 vtss\_packet\_rx\_conf\_t, 129  
     grp\_map, 130  
     map, 129  
     queue, 129  
     reg, 129  
 vtss\_packet\_rx\_frame\_discard  
     vtss\_packet\_api.h, 539  
 vtss\_packet\_rx\_frame\_get  
     vtss\_packet\_api.h, 538  
 vtss\_packet\_rx\_frame\_get\_raw  
     vtss\_packet\_api.h, 539  
 vtss\_packet\_rx\_grp\_t  
     types.h, 375  
 vtss\_packet\_rx\_hdr\_decode  
     vtss\_packet\_api.h, 543  
 vtss\_packet\_rx\_header\_t, 130  
     arrived\_tagged, 131  
     learn, 131  
     length, 130  
     port\_no, 131  
     queue\_mask, 131  
     tag, 131  
 vtss\_packet\_rx\_hints\_t  
     vtss\_packet\_api.h, 533  
 vtss\_packet\_rx\_info\_t, 132  
     acl\_hit, 137  
     acl\_idx, 137  
     cos, 136  
     glag\_no, 134  
     hints, 133  
     hw\_tstamp, 138  
     hw\_tstamp\_decoded, 139  
     isdx, 140  
     length, 133  
     oam\_info, 140  
     oam\_info\_decoded, 140  
     port\_no, 133  
     sflow\_port\_no, 139  
     sflow\_type, 139  
     stripped\_tag, 135  
     sw\_tstamp, 137  
     tag, 135  
     tag\_type, 134  
     tstamp\_id, 138  
     tstamp\_id\_decoded, 138  
     xtr\_qu\_mask, 136  
 vtss\_packet\_rx\_meta\_t, 141  
     chip\_no, 142  
     etype, 143  
     fcs, 143  
     length, 144  
     no\_wait, 142  
     sw\_tstamp, 144  
     xtr\_qu, 142  
 vtss\_packet\_rx\_port\_conf\_get  
     vtss\_packet\_api.h, 536  
 vtss\_packet\_rx\_port\_conf\_set  
     vtss\_packet\_api.h, 538

vtss\_packet\_rx\_port\_conf\_t, 145  
  bpdu\_reg, 146  
  garp\_reg, 146  
vtss\_packet\_rx\_queue\_conf\_t, 146  
  npi, 147  
  size, 146  
vtss\_packet\_rx\_queue\_map\_t, 147  
  bpdu\_queue, 148  
  garp\_queue, 148  
  igmp\_queue, 148  
  ipmc\_ctrl\_queue, 148  
  learn\_queue, 148  
  lrn\_all\_queue, 149  
  mac\_vid\_queue, 148  
  sflow\_queue, 149  
  stack\_queue, 149  
vtss\_packet\_rx\_queue\_npi\_conf\_t, 149  
  enable, 150  
vtss\_packet\_rx\_reg\_t, 150  
  bpdu\_cpu\_only, 151  
  garp\_cpu\_only, 151  
  igmp\_cpu\_only, 151  
  ipmc\_ctrl\_cpu\_copy, 151  
  mld\_cpu\_only, 151  
vtss\_packet\_tx\_frame  
  vtss\_packet\_api.h, 545  
vtss\_packet\_tx\_frame\_port  
  vtss\_packet\_api.h, 540  
vtss\_packet\_tx\_frame\_port\_vlan  
  vtss\_packet\_api.h, 540  
vtss\_packet\_tx\_frame\_vlan  
  vtss\_packet\_api.h, 541  
vtss\_packet\_tx\_grp\_t  
  types.h, 375  
vtss\_packet\_tx\_hdr\_compile  
  vtss\_packet\_api.h, 545  
vtss\_packet\_tx\_hdr\_encode  
  vtss\_packet\_api.h, 543  
vtss\_packet\_tx\_ifh\_t, 152  
  ifh, 152  
  length, 152  
vtss\_packet\_tx\_info\_init  
  vtss\_packet\_api.h, 546  
vtss\_packet\_tx\_info\_t, 153  
  aggr\_code, 156  
  cos, 156  
  dp, 160  
  dst\_port\_mask, 154  
  frm\_len, 155  
  isdx, 159  
  isdx\_dont\_use, 160  
  latch\_timestamp, 158  
  masquerade\_port, 161  
  oam\_type, 159  
  pdu\_offset, 161  
  ptp\_action, 157  
  ptp\_id, 157  
  ptp\_timestamp, 158  
switch\_frm, 154  
tag, 155  
vtss\_packet\_tx\_vstax\_t  
  vtss\_packet\_api.h, 534  
vtss\_phy\_aneg\_t, 162  
  asymmetric\_pause, 164  
  speed\_100m\_fdx, 163  
  speed\_100m\_hdx, 163  
  speed\_10m\_fdx, 163  
  speed\_10m\_hdx, 163  
  speed\_1g\_fdx, 163  
  speed\_1g\_hdx, 163  
  symmetric\_pause, 164  
  tx\_remote\_fault, 164  
vtss\_phy\_api.h  
  lb\_type, 578  
  MAX\_CFG\_BUF\_SIZE, 558  
  MAX\_STAT\_BUF\_SIZE, 558  
  MAX\_WOL\_MAC\_ADDR\_SIZE, 569  
  MAX\_WOL\_PASSWD\_SIZE, 569  
  MIN\_WOL\_PASSWD\_SIZE, 569  
  rgmii\_skew\_delay\_psec\_t, 572  
  VTSS\_PHY\_ACTIPHY\_PWR, 559  
  VTSS\_PHY\_LINK\_AMS\_EV, 564  
  VTSS\_PHY\_LINK\_AUTO\_NEG\_COMPLETE\_EV, 564  
  VTSS\_PHY\_LINK\_AUTO\_NEG\_ERROR\_EV, 564  
  VTSS\_PHY\_LINK\_DOWN\_PWR, 559  
  VTSS\_PHY\_LINK\_EXT\_EEE\_LINKFAIL\_EV, 567  
  VTSS\_PHY\_LINK\_EXT\_EEE\_WAIT\_RX\_TQ\_EV, 567  
  VTSS\_PHY\_LINK\_EXT\_EEE\_WAIT\_TS\_EV, 567  
  VTSS\_PHY\_LINK\_EXT\_EEE\_WAKE\_ERR\_EV, 567  
  VTSS\_PHY\_LINK\_EXT\_MACSEC\_EGRESS\_EV, 568  
  VTSS\_PHY\_LINK\_EXT\_MACSEC\_FC\_BUFF\_EV, 568  
  VTSS\_PHY\_LINK\_EXT\_MACSEC\_HOST\_MAC\_EV, 568  
  VTSS\_PHY\_LINK\_EXT\_MACSEC\_INGRESS\_EV, 568  
  VTSS\_PHY\_LINK\_EXT\_MACSEC\_LINE\_MAC\_EV, 568  
  VTSS\_PHY\_LINK\_EXT\_MEM\_INT\_RING\_EV, 569  
  VTSS\_PHY\_LINK\_EXT\_RR\_SW\_COMPL\_EV, 567  
  VTSS\_PHY\_LINK\_EXTENDED\_REG\_INT\_EV, 566  
  VTSS\_PHY\_LINK\_FALSE\_CARRIER\_INT\_EV, 565  
  VTSS\_PHY\_LINK\_FDX\_STATE\_CHANGE\_EV, 564  
  VTSS\_PHY\_LINK\_FFAIL\_EV, 563  
  VTSS\_PHY\_LINK\_INLINE\_POW\_DEV\_DETECT\_EV, 565  
  VTSS\_PHY\_LINK\_LINK\_SPEED\_DS\_DETECTEV,

\_EV, 566  
 VTSS\_PHY\_LINK\_LOS\_EV, 563  
 VTSS\_PHY\_LINK\_MASTER\_SLAVE\_RES\_ER←  
     R\_EV, 566  
 VTSS\_PHY\_LINK\_RX\_ER\_INT\_EV, 566  
 VTSS\_PHY\_LINK\_RX\_FIFO\_OVERFLOW\_INT←  
     \_EV, 565  
 VTSS\_PHY\_LINK\_SPEED\_STATE\_CHANGE←  
     EV, 564  
 VTSS\_PHY\_LINK\_SYMBOL\_ERR\_INT\_EV, 565  
 VTSS\_PHY\_LINK\_TX\_FIFO\_OVERFLOW\_INT←  
     \_EV, 565  
 VTSS\_PHY\_LINK\_UP\_FULL\_PWR, 559  
 VTSS\_PHY\_LINK\_WAKE\_ON\_LAN\_INT\_EV, 566  
 VTSS\_PHY\_PAGE\_0x2DAF, 562  
 VTSS\_PHY\_PAGE\_1588, 561  
 VTSS\_PHY\_PAGE\_EXTENDED\_2, 560  
 VTSS\_PHY\_PAGE\_EXTENDED\_3, 561  
 VTSS\_PHY\_PAGE\_EXTENDED\_4, 561  
 VTSS\_PHY\_PAGE\_EXTENDED, 560  
 VTSS\_PHY\_PAGE\_GPIO, 561  
 VTSS\_PHY\_PAGE\_MACSEC, 561  
 VTSS\_PHY\_PAGE\_STANDARD, 560  
 VTSS\_PHY\_PAGE\_TEST, 562  
 VTSS\_PHY\_PAGE\_TR, 562  
 VTSS\_PHY\_POWER\_ACTIPHY\_BIT, 558  
 VTSS\_PHY\_POWER\_DYNAMIC\_BIT, 559  
 VTSS\_PHY\_RECov\_CLK1, 559  
 VTSS\_PHY\_RECov\_CLK2, 560  
 VTSS\_PHY\_RECov\_CLK\_NUM, 560  
 VTSS\_PHY\_REG\_EXTENDED, 562  
 VTSS\_PHY\_REG\_GPIO, 563  
 VTSS\_PHY\_REG\_STANDARD, 562  
 VTSS\_PHY\_REG\_TEST, 563  
 VTSS\_PHY\_REG\_TR, 563  
 vga\_adc\_debug, 598  
 vtss\_fefi\_mode\_t, 579  
 vtss\_phy\_atom12\_patch\_settings\_get, 608  
 vtss\_phy\_cfg\_ib\_cterm, 608  
 vtss\_phy\_cfg\_ob\_post0, 607  
 vtss\_phy\_chip\_temp\_get, 581  
 vtss\_phy\_chip\_temp\_init, 582  
 vtss\_phy\_cl37\_lp\_abil\_get, 583  
 vtss\_phy\_clk\_source\_t, 577  
 vtss\_phy\_clk\_squelch, 577  
 vtss\_phy\_clock\_conf\_get, 587  
 vtss\_phy\_clock\_conf\_set, 587  
 vtss\_phy\_coma\_mode\_disable, 597  
 vtss\_phy\_coma\_mode\_enable, 598  
 vtss\_phy\_conf\_1g\_get, 584  
 vtss\_phy\_conf\_1g\_set, 585  
 vtss\_phy\_conf\_get, 582  
 vtss\_phy\_conf\_set, 583  
 vtss\_phy\_csr\_rd, 603  
 vtss\_phy\_csr\_wr, 603  
 vtss\_phy\_debug\_phyinfo\_print, 611  
 vtss\_phy\_debug\_reddump\_print, 613  
 vtss\_phy\_debug\_register\_dump, 611  
 vtss\_phy\_debug\_stat\_print, 610  
 vtss\_phy\_detect\_base\_ports, 612  
 vtss\_phy\_do\_page\_chk\_get, 606  
 vtss\_phy\_do\_page\_chk\_set, 605  
 vtss\_phy\_eee\_conf\_get, 599  
 vtss\_phy\_eee\_conf\_set, 600  
 vtss\_phy\_eee\_ena, 599  
 vtss\_phy\_eee\_link\_partner\_advertisements\_get,  
     601  
 vtss\_phy\_eee\_power\_save\_state\_get, 600  
 vtss\_phy\_enhanced\_led\_control\_init, 596  
 vtss\_phy\_enhanced\_led\_control\_init\_get, 597  
 vtss\_phy\_event\_enable\_get, 601  
 vtss\_phy\_event\_enable\_set, 601  
 vtss\_phy\_event\_poll, 602  
 vtss\_phy\_ext\_connector\_loopback, 612  
 vtss\_phy\_ext\_event\_poll, 623  
 vtss\_phy\_fast\_link\_fail\_t, 573  
 vtss\_phy\_fefi\_detect, 621  
 vtss\_phy\_fefi\_get, 620  
 vtss\_phy\_fefi\_set, 620  
 vtss\_phy\_flowcontrol\_decode\_status, 609  
 vtss\_phy\_forced\_reset\_t, 572  
 vtss\_phy\_freq\_t, 577  
 vtss\_phy\_gpio\_get, 593  
 vtss\_phy\_gpio\_mode, 593  
 vtss\_phy\_gpio\_mode\_t, 578  
 vtss\_phy\_gpio\_set, 594  
 vtss\_phy\_i2c\_read, 588  
 vtss\_phy\_i2c\_write, 589  
 vtss\_phy\_id\_get, 584  
 vtss\_phy\_is\_8051\_crc\_ok, 607  
 vtss\_phy\_is\_viper\_revB, 623  
 vtss\_phy\_lcpll\_status\_get, 617  
 vtss\_phy\_led\_intensity, 570  
 vtss\_phy\_led\_intensity\_get, 596  
 vtss\_phy\_led\_intensity\_set, 595  
 vtss\_phy\_led\_mode\_set, 595  
 vtss\_phy\_led\_mode\_t, 570  
 vtss\_phy\_loopback\_get, 606  
 vtss\_phy\_loopback\_set, 606  
 vtss\_phy\_mac\_media\_inhibit\_odd\_start, 619  
 vtss\_phy\_mac\_serdes\_pcs\_sgmii\_pre, 574  
 vtss\_phy\_macsec\_csr\_sd6g\_rd, 624  
 vtss\_phy\_macsec\_csr\_sd6g\_wr, 624  
 vtss\_phy\_mdi\_t, 572  
 vtss\_phy\_media\_force\_ams\_sel\_t, 576  
 vtss\_phy\_media\_interface\_t, 571  
 vtss\_phy\_media\_rem\_fault\_t, 576  
 vtss\_phy\_mmd\_read, 590  
 vtss\_phy\_mmd\_write, 591  
 vtss\_phy\_mode\_t, 573  
 vtss\_phy\_mse\_1000m\_get, 622  
 vtss\_phy\_mse\_100m\_get, 621  
 vtss\_phy\_patch\_settings\_get, 615  
 vtss\_phy\_pkt\_mode\_t, 573  
 vtss\_phy\_port\_eee\_capable, 598  
 vtss\_phy\_post\_reset, 580

vtss\_phy\_power\_conf\_get, 586  
vtss\_phy\_power\_conf\_set, 586  
vtss\_phy\_power\_status\_get, 586  
vtss\_phy\_pre\_reset, 579  
vtss\_phy\_pre\_system\_reset, 580  
vtss\_phy\_read, 589  
vtss\_phy\_read\_page, 590  
vtss\_phy\_read\_tr\_addr, 622  
vtss\_phy\_recov\_clk\_t, 570  
vtss\_phy\_reg\_decode\_status, 609  
vtss\_phy\_reset, 580  
vtss\_phy\_reset\_get, 581  
vtss\_phy\_reset\_lcpll, 617  
vtss\_phy\_sd6g\_mac\_serdes\_conf, 625  
vtss\_phy\_sd6g\_ob\_lev\_rd, 619  
vtss\_phy\_sd6g\_ob\_lev\_wr, 619  
vtss\_phy\_sd6g\_ob\_post\_rd, 618  
vtss\_phy\_sd6g\_ob\_post\_wr, 618  
vtss\_phy\_serdes1g\_rcpll\_status\_get, 616  
vtss\_phy\_serdes6g\_rcpll\_status\_get, 616  
vtss\_phy\_serdes\_fmedia\_loopback, 613  
vtss\_phy\_serdes\_sgmii\_loopback, 612  
vtss\_phy\_sigdet\_polarity\_t, 574  
vtss\_phy\_statistic\_get, 605  
vtss\_phy\_status\_1g\_get, 585  
vtss\_phy\_status\_get, 583  
vtss\_phy\_status\_inst\_poll, 624  
vtss\_phy\_unidirectional\_t, 574  
vtss\_phy\_veriphy\_get, 595  
vtss\_phy\_veriphy\_start, 594  
vtss\_phy\_warm\_start\_failed\_get, 610  
vtss\_phy\_wol\_conf\_get, 614  
vtss\_phy\_wol\_conf\_set, 615  
vtss\_phy\_wol\_enable, 614  
vtss\_phy\_write, 591  
vtss\_phy\_write\_masked, 592  
vtss\_phy\_write\_masked\_page, 592  
vtss\_squelch\_workaround, 602  
vtss\_wol\_passwd\_len\_type\_t, 579  
vtss\_phy\_atom12\_patch\_settings\_get  
    vtss\_phy\_api.h, 608  
vtss\_phy\_cfg\_ib\_cterm  
    vtss\_phy\_api.h, 608  
vtss\_phy\_cfg\_ob\_post0  
    vtss\_phy\_api.h, 607  
vtss\_phy\_chip\_temp\_get  
    vtss\_phy\_api.h, 581  
vtss\_phy\_chip\_temp\_init  
    vtss\_phy\_api.h, 582  
vtss\_phy\_cl37\_lp\_abil\_get  
    vtss\_phy\_api.h, 583  
vtss\_phy\_clk\_source\_t  
    vtss\_phy\_api.h, 577  
vtss\_phy\_clk\_squelch  
    vtss\_phy\_api.h, 577  
vtss\_phy\_clock\_conf\_get  
    vtss\_phy\_api.h, 587  
vtss\_phy\_clock\_conf\_set  
    vtss\_phy\_api.h, 587  
vtss\_phy\_coma\_mode\_disable  
    vtss\_phy\_api.h, 597  
vtss\_phy\_coma\_mode\_enable  
    vtss\_phy\_api.h, 598  
vtss\_phy\_conf\_1g\_get  
    vtss\_phy\_api.h, 584  
vtss\_phy\_conf\_1g\_set  
    vtss\_phy\_api.h, 585  
vtss\_phy\_conf\_1g\_t, 166  
    cfg, 166  
    master, 166  
    val, 166  
vtss\_phy\_conf\_get  
    vtss\_phy\_api.h, 582  
vtss\_phy\_conf\_set  
    vtss\_phy\_api.h, 583  
vtss\_phy\_conf\_t, 167  
    aneg, 168  
    flf, 168  
    force\_ams\_sel, 169  
    forced, 167  
    mac\_if\_pcs, 169  
    mdi, 168  
    media\_if\_pcs, 169  
    mode, 167  
    sigdet, 168  
    skip\_coma, 169  
    unidir, 168  
vtss\_phy\_csr\_rd  
    vtss\_phy\_api.h, 603  
vtss\_phy\_csr\_wr  
    vtss\_phy\_api.h, 603  
vtss\_phy\_debug\_phyinfo\_print  
    vtss\_phy\_api.h, 611  
vtss\_phy\_debug\_regdump\_print  
    vtss\_phy\_api.h, 613  
vtss\_phy\_debug\_register\_dump  
    vtss\_phy\_api.h, 611  
vtss\_phy\_debug\_stat\_print  
    vtss\_phy\_api.h, 610  
vtss\_phy\_detect\_base\_ports  
    vtss\_phy\_api.h, 612  
vtss\_phy\_do\_page\_chk\_get  
    vtss\_phy\_api.h, 606  
vtss\_phy\_do\_page\_chk\_set  
    vtss\_phy\_api.h, 605  
vtss\_phy\_eee\_conf\_get  
    vtss\_phy\_api.h, 599  
vtss\_phy\_eee\_conf\_set  
    vtss\_phy\_api.h, 600  
vtss\_phy\_eee\_conf\_t, 170  
    eee\_ena\_phy, 170  
    eee\_mode, 170

vtss\_phy\_eee\_ena  
     vtss\_phy\_api.h, 599  
 vtss\_phy\_eee\_link\_partner\_advertisements\_get  
     vtss\_phy\_api.h, 601  
 vtss\_phy\_eee\_power\_save\_state\_get  
     vtss\_phy\_api.h, 600  
 vtss\_phy\_enhanced\_led\_control\_init  
     vtss\_phy\_api.h, 596  
 vtss\_phy\_enhanced\_led\_control\_init\_get  
     vtss\_phy\_api.h, 597  
 vtss\_phy\_enhanced\_led\_control\_t, 170  
     ser\_led\_frame\_rate, 171  
     ser\_led\_output\_1, 171  
     ser\_led\_output\_2, 171  
     ser\_led\_select, 171  
 vtss\_phy\_event\_enable\_get  
     vtss\_phy\_api.h, 601  
 vtss\_phy\_event\_enable\_set  
     vtss\_phy\_api.h, 601  
 vtss\_phy\_event\_poll  
     vtss\_phy\_api.h, 602  
 vtss\_phy\_ext\_connector\_loopback  
     vtss\_phy\_api.h, 612  
 vtss\_phy\_ext\_event\_poll  
     vtss\_phy\_api.h, 623  
 vtss\_phy\_fast\_link\_fail\_t  
     vtss\_phy\_api.h, 573  
 vtss\_phy\_fefi\_detect  
     vtss\_phy\_api.h, 621  
 vtss\_phy\_fefi\_get  
     vtss\_phy\_api.h, 620  
 vtss\_phy\_fefi\_set  
     vtss\_phy\_api.h, 620  
 vtss\_phy\_flowcontrol\_decode\_status  
     vtss\_phy\_api.h, 609  
 vtss\_phy\_forced\_reset\_t  
     vtss\_phy\_api.h, 572  
 vtss\_phy\_forced\_t, 172  
     fdx, 172  
     speed, 172  
 vtss\_phy\_freq\_t  
     vtss\_phy\_api.h, 577  
 vtss\_phy\_gpio\_get  
     vtss\_phy\_api.h, 593  
 vtss\_phy\_gpio\_mode  
     vtss\_phy\_api.h, 593  
 vtss\_phy\_gpio\_mode\_t  
     vtss\_phy\_api.h, 578  
 vtss\_phy\_gpio\_set  
     vtss\_phy\_api.h, 594  
 vtss\_phy\_i2c\_read  
     vtss\_phy\_api.h, 588  
 vtss\_phy\_i2c\_write  
     vtss\_phy\_api.h, 589  
 vtss\_phy\_id\_get  
     vtss\_phy\_api.h, 584  
 vtss\_phy\_is\_8051\_crc\_ok  
     vtss\_phy\_api.h, 607  
 vtss\_phy\_is\_viper\_revB  
     vtss\_phy\_api.h, 623  
 vtss\_phy\_lcpll\_status\_get  
     vtss\_phy\_api.h, 617  
 vtss\_phy\_led\_intensity  
     vtss\_phy\_api.h, 570  
 vtss\_phy\_led\_intensity\_get  
     vtss\_phy\_api.h, 596  
 vtss\_phy\_led\_intensity\_set  
     vtss\_phy\_api.h, 595  
 vtss\_phy\_led\_mode\_select\_t, 173  
     mode, 173  
     number, 173  
 vtss\_phy\_led\_mode\_set  
     vtss\_phy\_api.h, 595  
 vtss\_phy\_led\_mode\_t  
     vtss\_phy\_api.h, 570  
 vtss\_phy\_loopback\_get  
     vtss\_phy\_api.h, 606  
 vtss\_phy\_loopback\_set  
     vtss\_phy\_api.h, 606  
 vtss\_phy\_loopback\_t, 174  
     connector\_enable, 175  
     far\_end\_enable, 174  
     mac\_serdes\_equipment\_enable, 175  
     mac\_serdes\_facility\_enable, 175  
     mac\_serdes\_input\_enable, 175  
     media\_serdes\_equipment\_enable, 176  
     media\_serdes\_facility\_enable, 176  
     media\_serdes\_input\_enable, 175  
     near\_end\_enable, 174  
 vtss\_phy\_mac\_media\_inhibit\_odd\_start  
     vtss\_phy\_api.h, 619  
 vtss\_phy\_mac\_serd\_pcs\_cntl\_t, 176  
     aneg\_restart, 177  
     disable, 177  
     fast\_link\_stat\_ena, 179  
     force\_adv\_ability, 177  
     inhibit\_odd\_start, 179  
     pd\_enable, 177  
     restart, 177  
     serdes\_aneg\_ena, 178  
     serdes\_pol\_inv\_in, 178  
     serdes\_pol\_inv\_out, 178  
     sgmii\_in\_pre, 178  
     sgmii\_out\_pre, 178  
 vtss\_phy\_mac\_serd\_pcs\_sgmii\_pre  
     vtss\_phy\_api.h, 574  
 vtss\_phy\_macsec\_csr\_sd6g\_rd  
     vtss\_phy\_api.h, 624  
 vtss\_phy\_macsec\_csr\_sd6g\_wr  
     vtss\_phy\_api.h, 624  
 vtss\_phy\_mdi\_t  
     vtss\_phy\_api.h, 572  
 vtss\_phy\_media\_force\_ams\_sel\_t  
     vtss\_phy\_api.h, 576  
 vtss\_phy\_media\_interface\_t  
     vtss\_phy\_api.h, 571

vtss\_phy\_media\_rem\_fault\_t  
vtss\_phy\_api.h, 576

vtss\_phy\_media\_serdes\_pcs\_cntl\_t, 179  
aneg\_pd\_detect, 180  
force\_adv\_ability, 180  
force\_fefi, 181  
force\_fefi\_value, 181  
force\_hls, 181  
inhibit\_odd\_start, 181  
remote\_fault, 180  
serdes\_pol\_inv\_in, 180  
serdes\_pol\_inv\_out, 180

vtss\_phy\_mmd\_read  
vtss\_phy\_api.h, 590

vtss\_phy\_mmd\_write  
vtss\_phy\_api.h, 591

vtss\_phy\_mode\_t  
vtss\_phy\_api.h, 573

vtss\_phy\_mse\_1000m\_get  
vtss\_phy\_api.h, 622

vtss\_phy\_mse\_100m\_get  
vtss\_phy\_api.h, 621

vtss\_phy\_patch\_settings\_get  
vtss\_phy\_api.h, 615

vtss\_phy\_pkt\_mode\_t  
vtss\_phy\_api.h, 573

vtss\_phy\_port\_eee\_capable  
vtss\_phy\_api.h, 598

vtss\_phy\_post\_reset  
vtss\_phy\_api.h, 580

vtss\_phy\_power\_conf\_get  
vtss\_phy\_api.h, 586

vtss\_phy\_power\_conf\_set  
vtss\_phy\_api.h, 586

vtss\_phy\_power\_conf\_t, 182  
mode, 182

vtss\_phy\_power\_mode\_t  
phy.h, 333

vtss\_phy\_power\_status\_get  
vtss\_phy\_api.h, 586

vtss\_phy\_power\_status\_t, 182  
level, 183

vtss\_phy\_pre\_reset  
vtss\_phy\_api.h, 579

vtss\_phy\_pre\_system\_reset  
vtss\_phy\_api.h, 580

vtss\_phy\_read  
vtss\_phy\_api.h, 589

vtss\_phy\_read\_page  
vtss\_phy\_api.h, 590

vtss\_phy\_read\_tr\_addr  
vtss\_phy\_api.h, 622

vtss\_phy\_recov\_clk\_t  
vtss\_phy\_api.h, 570

vtss\_phy\_reg\_decode\_status  
vtss\_phy\_api.h, 609

vtss\_phy\_reset  
vtss\_phy\_api.h, 580

vtss\_phy\_reset\_conf\_t, 183  
force, 184  
i\_cpu\_en, 185  
mac\_if, 184  
media\_if, 184  
pkt\_mode, 184  
rgmii, 184  
tbi, 184

vtss\_phy\_reset\_get  
vtss\_phy\_api.h, 581

vtss\_phy\_reset\_lcpll  
vtss\_phy\_api.h, 617

vtss\_phy\_rgmii\_conf\_t, 185  
rx\_clk\_skew\_ps, 185  
tx\_clk\_skew\_ps, 186

vtss\_phy\_sd6g\_mac\_serdes\_conf  
vtss\_phy\_api.h, 625

vtss\_phy\_sd6g\_ob\_lev\_rd  
vtss\_phy\_api.h, 619

vtss\_phy\_sd6g\_ob\_lev\_wr  
vtss\_phy\_api.h, 619

vtss\_phy\_sd6g\_ob\_post\_rd  
vtss\_phy\_api.h, 618

vtss\_phy\_sd6g\_ob\_post\_wr  
vtss\_phy\_api.h, 618

vtss\_phy\_serdes1g\_rcpll\_status\_get  
vtss\_phy\_api.h, 616

vtss\_phy\_serdes6g\_rcpll\_status\_get  
vtss\_phy\_api.h, 616

vtss\_phy\_serdes\_fmedia\_loopback  
vtss\_phy\_api.h, 613

vtss\_phy\_serdes\_sgmii\_loopback  
vtss\_phy\_api.h, 612

vtss\_phy\_sigdet\_polarity\_t  
vtss\_phy\_api.h, 574

vtss\_phy\_statistic\_get  
vtss\_phy\_api.h, 605

vtss\_phy\_statistic\_t, 186  
cu\_bad, 187  
cu\_good, 186  
media\_mac\_serdes\_crc, 188  
media\_mac\_serdes\_good, 187  
rx\_err\_cnt\_base\_tx, 187  
serdes\_tx\_bad, 187  
serdes\_tx\_good, 187

vtss\_phy\_status\_1g\_get  
vtss\_phy\_api.h, 585

vtss\_phy\_status\_1g\_t, 188  
master, 189  
master\_cfg\_fault, 188

vtss\_phy\_status\_get  
vtss\_phy\_api.h, 583

vtss\_phy\_status\_inst\_poll  
vtss\_phy\_api.h, 624

vtss\_phy\_tbi\_conf\_t, 189  
aneg\_enable, 189

vtss\_phy\_type\_t, 190  
base\_port\_no, 191

channel\_id, 191  
 part\_number, 190  
 phy\_api\_base\_no, 191  
 port\_cnt, 191  
 revision, 190  
 vtss\_phy\_unidirectional\_t  
     vtss\_phy\_api.h, 574  
 vtss\_phy\_veriphy\_get  
     vtss\_phy\_api.h, 595  
 vtss\_phy\_veriphy\_result\_t, 192  
     length, 192  
     link, 192  
     status, 192  
 vtss\_phy\_veriphy\_start  
     vtss\_phy\_api.h, 594  
 vtss\_phy\_veriphy\_status\_t  
     phy.h, 334  
 vtss\_phy\_warm\_start\_failed\_get  
     vtss\_phy\_api.h, 610  
 vtss\_phy\_wol\_conf\_get  
     vtss\_phy\_api.h, 614  
 vtss\_phy\_wol\_conf\_set  
     vtss\_phy\_api.h, 615  
 vtss\_phy\_wol\_conf\_t, 193  
     magic\_pkt\_cnt, 194  
     secure\_on\_enable, 193  
     wol\_mac, 193  
     wol\_pass, 194  
     wol\_passwd\_len, 194  
 vtss\_phy\_wol\_enable  
     vtss\_phy\_api.h, 614  
 vtss\_phy\_write  
     vtss\_phy\_api.h, 591  
 vtss\_phy\_write\_masked  
     vtss\_phy\_api.h, 592  
 vtss\_phy\_write\_masked\_page  
     vtss\_phy\_api.h, 592  
 vtss\_pi\_conf\_t, 194  
     cs\_wait\_ns, 195  
     use\_extended\_bus\_cycle, 195  
     width, 195  
 vtss\_policer\_ext\_t, 196  
     flow\_control, 196  
     frame\_rate, 196  
 vtss\_policer\_t, 196  
     level, 197  
     rate, 197  
 vtss\_policer\_type\_t  
     types.h, 380  
 vtss\_poll\_1sec  
     vtss\_misc\_api.h, 487  
 vtss\_port\_api.h  
     CHIP\_PORT\_UNUSED, 628  
     VTSS\_FRAME\_GAP\_DEFAULT, 628  
     VTSS\_MAX\_FRAME\_LENGTH\_MAX, 629  
     VTSS\_MAX\_FRAME\_LENGTH\_STANDARD, 628  
     vtss\_internal\_bw\_t, 629  
     vtss\_miim\_controller\_t, 629  
         vtss\_miim\_read, 637  
         vtss\_miim\_write, 637  
         vtss\_port\_basic\_counters\_get, 636  
         vtss\_port\_clause\_37\_control\_get, 632  
         vtss\_port\_clause\_37\_control\_set, 633  
         vtss\_port\_clause\_37\_remote\_fault\_t, 630  
         vtss\_port\_conf\_get, 633  
         vtss\_port\_conf\_set, 633  
         vtss\_port\_counters\_clear, 635  
         vtss\_port\_counters\_get, 635  
         vtss\_port\_counters\_update, 634  
         vtss\_port\_forward\_state\_get, 636  
         vtss\_port\_forward\_state\_set, 636  
         vtss\_port\_forward\_t, 631  
         vtss\_port\_loop\_t, 631  
         vtss\_port\_map\_get, 632  
         vtss\_port\_map\_set, 631  
         vtss\_port\_max\_tags\_t, 630  
         vtss\_port\_status\_get, 634  
     vtss\_port\_basic\_counters\_get  
         vtss\_port\_api.h, 636  
     vtss\_port\_bridge\_counters\_t, 197  
         dot1dTpPortInDiscards, 198  
     vtss\_port\_clause\_37\_adv\_t, 198  
         acknowledge, 199  
         asymmetric\_pause, 199  
         fdx, 199  
         hdx, 199  
         next\_page, 200  
         remote\_fault, 199  
         symmetric\_pause, 199  
     vtss\_port\_clause\_37\_control\_get  
         vtss\_port\_api.h, 632  
     vtss\_port\_clause\_37\_control\_set  
         vtss\_port\_api.h, 633  
     vtss\_port\_clause\_37\_control\_t, 200  
         advertisement, 201  
         enable, 200  
     vtss\_port\_clause\_37\_remote\_fault\_t  
         vtss\_port\_api.h, 630  
     vtss\_port\_conf\_get  
         vtss\_port\_api.h, 633  
     vtss\_port\_conf\_set  
         vtss\_port\_api.h, 633  
     vtss\_port\_conf\_t, 201  
         exc\_col\_cont, 204  
         fdx, 203  
         flow\_control, 203  
         frame\_gaps, 202  
         frame\_length\_chk, 204  
         if\_type, 202  
         loop, 205  
         max\_frame\_length, 203  
         max\_tags, 204  
         power\_down, 203  
         sd\_active\_high, 202  
         sd\_enable, 202  
         sd\_internal, 202

serdes, 205  
speed, 203  
xaui\_rx\_lane\_flip, 204  
xaui\_tx\_lane\_flip, 204  
vtss\_port\_counters\_clear  
    vtss\_port\_api.h, 635  
vtss\_port\_counters\_get  
    vtss\_port\_api.h, 635  
vtss\_port\_counters\_t, 205  
    bridge, 206  
    ethernet\_like, 206  
    evc, 207  
    if\_group, 206  
    prop, 206  
    rmon, 206  
vtss\_port\_counters\_update  
    vtss\_port\_api.h, 634  
vtss\_port\_ethernet\_like\_counters\_t, 207  
    dot3InPauseFrames, 207  
    dot3OutPauseFrames, 208  
vtss\_port\_evc\_counters\_t, 208  
    rx\_green, 208  
    rx\_green\_discard, 209  
    rx\_red, 209  
    rx\_yellow, 209  
    rx\_yellow\_discard, 209  
    tx\_green, 209  
    tx\_yellow, 210  
vtss\_port\_flow\_control\_conf\_t, 210  
    generate, 211  
    obey, 210  
    pfc, 211  
    smac, 211  
vtss\_port\_forward\_state\_get  
    vtss\_port\_api.h, 636  
vtss\_port\_forward\_state\_set  
    vtss\_port\_api.h, 636  
vtss\_port\_forward\_t  
    vtss\_port\_api.h, 631  
vtss\_port\_frame\_gaps\_t, 211  
    fdx\_gap, 212  
    hdx\_gap\_1, 212  
    hdx\_gap\_2, 212  
vtss\_port\_if\_group\_counters\_t, 213  
    ifInBroadcastPkts, 214  
    ifInDiscards, 214  
    ifInErrors, 214  
    ifInMulticastPkts, 213  
    ifInNUcastPkts, 214  
    ifInOctets, 213  
    ifInUcastPkts, 213  
    ifOutBroadcastPkts, 215  
    ifOutDiscards, 215  
    ifOutErrors, 216  
    ifOutMulticastPkts, 215  
    ifOutNUcastPkts, 215  
    ifOutOctets, 214  
    ifOutUcastPkts, 215  
vtss\_port\_interface\_t  
    types.h, 378  
vtss\_port\_loop\_t  
    vtss\_port\_api.h, 631  
vtss\_port\_map\_get  
    vtss\_port\_api.h, 632  
vtss\_port\_map\_set  
    vtss\_port\_api.h, 631  
vtss\_port\_map\_t, 216  
    chip\_no, 217  
    chip\_port, 216  
    miim\_addr, 217  
    miim\_chip\_no, 217  
    miim\_controller, 217  
vtss\_port\_max\_tags\_t  
    vtss\_port\_api.h, 630  
vtss\_port\_mux\_mode\_t  
    vtss\_init\_api.h, 406  
vtss\_port\_proprietary\_counters\_t, 218  
    rx\_prio, 218  
    tx\_prio, 218  
vtss\_port\_rmon\_counters\_t, 218  
    rx\_etherStatsBroadcastPkts, 220  
    rx\_etherStatsCRCAlignErrors, 220  
    rx\_etherStatsDropEvents, 219  
    rx\_etherStatsFragments, 221  
    rx\_etherStatsJabbers, 221  
    rx\_etherStatsMulticastPkts, 220  
    rx\_etherStatsOctets, 219  
    rx\_etherStatsOversizePkts, 221  
    rx\_etherStatsPkts, 220  
    rx\_etherStatsPkts1024to1518Octets, 222  
    rx\_etherStatsPkts128to255Octets, 222  
    rx\_etherStatsPkts1519toMaxOctets, 222  
    rx\_etherStatsPkts256to511Octets, 222  
    rx\_etherStatsPkts512to1023Octets, 222  
    rx\_etherStatsPkts64Octets, 221  
    rx\_etherStatsPkts65to127Octets, 221  
    rx\_etherStatsUndersizePkts, 220  
    tx\_etherStatsBroadcastPkts, 223  
    tx\_etherStatsCollisions, 224  
    tx\_etherStatsDropEvents, 223  
    tx\_etherStatsMulticastPkts, 223  
    tx\_etherStatsOctets, 223  
    tx\_etherStatsPkts, 223  
    tx\_etherStatsPkts1024to1518Octets, 225  
    tx\_etherStatsPkts128to255Octets, 224  
    tx\_etherStatsPkts1519toMaxOctets, 225  
    tx\_etherStatsPkts256to511Octets, 224  
    tx\_etherStatsPkts512to1023Octets, 225  
    tx\_etherStatsPkts64Octets, 224  
    tx\_etherStatsPkts65to127Octets, 224  
vtss\_port\_serdes\_conf\_t, 225  
    sfp\_dac, 226  
vtss\_port\_sgmii\_aneg\_t, 226  
    aneg\_complete, 228  
    fdx, 227  
    hdx, 227

link, 227  
 speed\_100M, 227  
 speed\_10M, 227  
 speed\_1G, 227  
 vtss\_port\_speed\_t  
     port.h, 346  
 vtss\_port\_state\_get  
     vtss\_l2\_api.h, 433  
 vtss\_port\_state\_set  
     vtss\_l2\_api.h, 433  
 vtss\_port\_status\_get  
     vtss\_port\_api.h, 634  
 vtss\_port\_status\_t, 228  
     aneg, 230  
     aneg\_complete, 229  
     copper, 230  
     fdx, 229  
     fiber, 230  
     link, 229  
     link\_down, 229  
     mdi\_cross, 230  
     remote\_fault, 229  
     speed, 229  
     unidirectional\_ability, 230  
 vtss\_ptp\_event\_enable  
     vtss\_misc\_api.h, 488  
 vtss\_ptp\_event\_poll  
     vtss\_misc\_api.h, 487  
 vtss\_pvlan\_port\_members\_get  
     vtss\_l2\_api.h, 446  
 vtss\_pvlan\_port\_members\_set  
     vtss\_l2\_api.h, 447  
 vtss\_qce\_action\_t, 231  
     dei, 233  
     dp, 232  
     dp\_enable, 232  
     dscp, 232  
     dscp\_enable, 232  
     pcp, 233  
     pcp\_dei\_enable, 232  
     policy\_no, 233  
     policy\_no\_enable, 233  
     prio, 231  
     prio\_enable, 231  
 vtss\_qce\_add  
     vtss\_qos\_api.h, 646  
 vtss\_qce\_del  
     vtss\_qos\_api.h, 647  
 vtss\_qce\_frame\_etype\_t, 234  
     data, 234  
     etype, 234  
 vtss\_qce\_frame\_ipv4\_t, 234  
     dport, 236  
     dscp, 235  
     fragment, 235  
     proto, 235  
     sip, 235  
     sport, 236  
 vtss\_qce\_frame\_ipv6\_t, 236  
     dport, 237  
     dscp, 237  
     proto, 237  
     sip, 237  
     sport, 237  
 vtss\_qce\_frame\_llc\_t, 238  
     data, 238  
 vtss\_qce\_frame\_snap\_t, 239  
     data, 239  
 vtss\_qce\_init  
     vtss\_qos\_api.h, 646  
 vtss\_qce\_key\_t, 239  
     etype, 240  
     frame, 241  
     ipv4, 241  
     ipv6, 241  
     llc, 241  
     mac, 240  
     port\_list, 240  
     snap, 241  
     tag, 240  
     type, 240  
 vtss\_qce\_mac\_t, 242  
     dmac\_bc, 242  
     dmac\_mc, 242  
     smac, 243  
 vtss\_qce\_t, 243  
     action, 244  
     id, 243  
     key, 244  
 vtss\_qce\_tag\_t, 244  
     dei, 245  
     pcp, 245  
     s\_tag, 245  
     tagged, 245  
     vid, 245  
 vtss\_qce\_type\_t  
     vtss\_qos\_api.h, 643  
 vtss\_qos\_api.h  
     VTSS\_PORT\_POLICER\_CPU\_QUEUES, 640  
     VTSS\_PORT\_POLICERS, 640  
     VTSS\_QCE\_ID\_LAST, 641  
     VTSS\_QCL\_ARRAY\_SIZE, 641  
     VTSS\_QCL\_ID\_END, 641  
     VTSS\_QCL\_ID\_START, 641  
     VTSS\_QCL\_IDS, 641  
     vtss\_dscp\_emode\_t, 642  
     vtss\_dscp\_mode\_t, 642  
     vtss\_mep\_policer\_conf\_get, 644  
     vtss\_mep\_policer\_conf\_set, 644  
     vtss\_qce\_add, 646  
     vtss\_qce\_del, 647  
     vtss\_qce\_init, 646  
     vtss\_qce\_type\_t, 643  
     vtss\_qos\_conf\_get, 643  
     vtss\_qos\_conf\_set, 644  
     vtss\_qos\_port\_conf\_get, 645

vtss\_qos\_port\_conf\_set, 645  
vtss\_tag\_remark\_mode\_t, 642  
vtss\_qos\_conf\_get  
    vtss\_qos\_api.h, 643  
vtss\_qos\_conf\_set  
    vtss\_qos\_api.h, 644  
vtss\_qos\_conf\_t, 246  
    dscp\_dp\_level\_map, 247  
    dscp\_qos\_class\_map, 247  
    dscp\_qos\_map, 247  
    dscp\_qos\_map\_dp1, 247  
    dscp\_remap, 248  
    dscp\_remap\_dp1, 248  
    dscp\_remark, 248  
    dscp\_translate\_map, 248  
    dscp\_trust, 247  
    policer\_bc, 250  
    policer\_bc\_frame\_rate, 250  
    policer\_bc\_mode, 250  
    policer\_mc, 249  
    policer\_mc\_frame\_rate, 249  
    policer\_mc\_mode, 249  
    policer\_uc, 248  
    policer\_uc\_frame\_rate, 249  
    policer\_uc\_mode, 249  
    prios, 246  
vtss\_qos\_port\_conf\_get  
    vtss\_qos\_api.h, 645  
vtss\_qos\_port\_conf\_set  
    vtss\_qos\_api.h, 645  
vtss\_qos\_port\_conf\_t, 250  
    default\_dei, 253  
    default\_dpl, 253  
    default\_prio, 252  
    dmac\_dip, 256  
    dp\_level\_map, 254  
    dscp\_class\_enable, 254  
    dscp\_emode, 254  
    dscp\_mode, 254  
    dscp\_translate, 254  
    dwrr\_enable, 256  
    excess\_enable, 252  
    policer\_ext\_port, 251  
    policer\_port, 251  
    policer\_queue, 252  
    qos\_class\_map, 253  
    queue\_pct, 256  
    shaper\_port, 252  
    shaper\_queue, 252  
    tag\_class\_enable, 253  
    tag\_default\_dei, 255  
    tag\_default\_pcp, 255  
    tag\_dei\_map, 255  
    tag\_pcp\_map, 255  
    tag\_remark\_mode, 255  
    usr\_prio, 253  
vtss\_rcpll\_status\_t, 256  
    cal\_error, 257  
        cal\_not\_done, 257  
        out\_of\_range, 257  
vtss\_reg\_read  
    vtss\_misc\_api.h, 485  
vtss\_reg\_read\_t  
    vtss\_init\_api.h, 399  
vtss\_reg\_write  
    vtss\_misc\_api.h, 485  
vtss\_reg\_write\_masked  
    vtss\_misc\_api.h, 486  
vtss\_reg\_write\_t  
    vtss\_init\_api.h, 400  
vtss\_restart\_conf\_end  
    vtss\_init\_api.h, 409  
vtss\_restart\_conf\_get  
    vtss\_init\_api.h, 409  
vtss\_restart\_conf\_set  
    vtss\_init\_api.h, 410  
vtss\_restart\_status\_get  
    vtss\_init\_api.h, 409  
vtss\_restart\_status\_t, 258  
    cur\_version, 258  
    prev\_version, 258  
    restart, 258  
vtss\_restart\_t  
    vtss\_init\_api.h, 406  
vtss\_routing\_entry\_t, 259  
    ipv4\_uc, 259  
    ipv6\_uc, 260  
    route, 260  
    type, 259  
    vlan, 260  
vtss\_rx\_master\_timestamp\_get  
    vtss\_ts\_api.h, 676  
vtss\_rx\_timestamp\_get  
    vtss\_ts\_api.h, 675  
vtss\_rx\_timestamp\_id\_release  
    vtss\_ts\_api.h, 676  
vtss\_secure\_on\_passwd\_t, 260  
    passwd, 261  
vtss\_security\_api.h  
    VTSS\_ACE\_ID\_LAST, 650  
    vtss\_ace\_add, 656  
    vtss\_ace\_bit\_t, 652  
    vtss\_ace\_counter\_clear, 657  
    vtss\_ace\_counter\_get, 657  
    vtss\_ace\_del, 656  
    vtss\_ace\_init, 655  
    vtss\_ace\_type\_t, 651  
    vtss\_acl\_policer\_conf\_get, 653  
    vtss\_acl\_policer\_conf\_set, 653  
    vtss\_acl\_port\_action\_t, 651  
    vtss\_acl\_port\_conf\_get, 654  
    vtss\_acl\_port\_conf\_set, 654  
    vtss\_acl\_port\_counter\_clear, 655  
    vtss\_acl\_port\_counter\_get, 655  
    vtss\_acl\_ptp\_action\_t, 651  
    vtss\_auth\_port\_state\_get, 652

vtss\_auth\_port\_state\_set, 652  
vtss\_auth\_state\_t, 650  
vtss\_serd़es\_macro\_conf\_t, 261  
ib\_cterm\_ena, 262  
qsgmii, 262  
serdes1g\_vdd, 261  
serdes6g\_vdd, 262  
vtss\_serd़es\_mode\_t  
    types.h, 379  
vtss\_sf़low\_port\_conf\_get  
    vtss\_l2\_api.h, 449  
vtss\_sf़low\_port\_conf\_set  
    vtss\_l2\_api.h, 449  
vtss\_sf़low\_port\_conf\_t, 262  
    sampling\_rate, 263  
    type, 263  
vtss\_sf़low\_sampling\_rate\_convert  
    vtss\_l2\_api.h, 450  
vtss\_sf़low\_type\_t  
    l2\_types.h, 315  
vtss\_sgpio\_bmode\_t  
    vtss\_misc\_api.h, 480  
vtss\_sgpio\_conf\_get  
    vtss\_misc\_api.h, 492  
vtss\_sgpio\_conf\_set  
    vtss\_misc\_api.h, 492  
vtss\_sgpio\_conf\_t, 263  
    bit\_count, 264  
    bmode, 264  
    port\_conf, 264  
vtss\_sgpio\_event\_enable  
    vtss\_misc\_api.h, 494  
vtss\_sgpio\_event\_poll  
    vtss\_misc\_api.h, 493  
vtss\_sgpio\_mode\_t  
    vtss\_misc\_api.h, 479  
vtss\_sgpio\_port\_conf\_t, 265  
    enabled, 265  
    int\_pol\_high, 265  
    mode, 265  
vtss\_sgpio\_port\_data\_t, 266  
    value, 266  
vtss\_sgpio\_read  
    vtss\_misc\_api.h, 493  
vtss\_shaper\_t, 267  
    level, 267  
    rate, 267  
vtss\_spi\_32bit\_read\_write\_t  
    vtss\_init\_api.h, 402  
vtss\_spi\_64bit\_read\_write\_t  
    vtss\_init\_api.h, 402  
vtss\_spi\_read\_write\_t  
    vtss\_init\_api.h, 401  
vtss\_squelch\_workaround  
    vtss\_phy\_api.h, 602  
vtss\_storm\_policer\_mode\_t  
    types.h, 380  
vtss\_stp\_port\_state\_get  
    vtss\_stp\_port\_state\_set  
        vtss\_l2\_api.h, 433  
    vtss\_stp\_state\_t  
        vtss\_l2\_api.h, 434  
    vtss\_sync\_api.h  
        VTSS\_SYNCE\_CLK\_MAX, 659  
        VTSS\_SYNCE\_CLK\_A, 659  
        VTSS\_SYNCE\_CLK\_B, 659  
        vtss\_sync\_clock\_in\_get, 661  
        vtss\_sync\_clock\_in\_set, 661  
        vtss\_sync\_clock\_out\_get, 660  
        vtss\_sync\_clock\_out\_set, 660  
        vtss\_sync\_divider\_t, 659  
    vtss\_sync\_clock\_in\_get  
        vtss\_sync\_api.h, 661  
    vtss\_sync\_clock\_in\_set  
        vtss\_sync\_api.h, 661  
vtss\_sync\_clock\_in\_t, 268  
    enable, 268  
    port\_no, 268  
    squersh, 268  
vtss\_sync\_clock\_out\_get  
    vtss\_sync\_api.h, 660  
vtss\_sync\_clock\_out\_set  
    vtss\_sync\_api.h, 660  
vtss\_sync\_clock\_out\_t, 269  
    divider, 269  
    enable, 269  
vtss\_sync\_divider\_t  
    vtss\_sync\_api.h, 659  
vtss\_tag\_remark\_mode\_t  
    vtss\_qos\_api.h, 642  
vtss\_tag\_type\_t  
    vtss\_packet\_api.h, 533  
vtss\_target\_type\_t  
    vtss\_init\_api.h, 405  
vtss\_tci\_t, 270  
    cfi, 270  
    tagprio, 271  
    vid, 270  
vtss\_temp\_sensor\_get  
    vtss\_misc\_api.h, 499  
vtss\_temp\_sensor\_init  
    vtss\_misc\_api.h, 498  
vtss\_timeofday\_t, 271  
    sec, 271, 272  
vtss\_timestamp\_age  
    vtss\_ts\_api.h, 677  
vtss\_timestamp\_t, 272  
    nanoseconds, 273  
    sec\_msb, 272  
    seconds, 273  
vtss\_tod\_get\_ns\_cnt  
    vtss\_misc\_api.h, 498  
vtss\_tod\_set\_ns\_cnt\_cb  
    vtss\_misc\_api.h, 498  
vtss\_trace\_conf\_get

vtss\_misc\_api.h, 481  
vtss\_trace\_conf\_set  
    vtss\_misc\_api.h, 481  
vtss\_trace\_conf\_t, 273  
    level, 274  
vtss\_trace\_group\_t  
    vtss\_misc\_api.h, 475  
vtss\_trace\_layer\_t  
    vtss\_misc\_api.h, 475  
vtss\_trace\_level\_t  
    vtss\_misc\_api.h, 476  
vtss\_ts\_adjtimer\_get  
    vtss\_ts\_api.h, 668  
vtss\_ts\_adjtimer\_one\_sec  
    vtss\_ts\_api.h, 666  
vtss\_ts\_adjtimer\_set  
    vtss\_ts\_api.h, 668  
vtss\_ts\_api.h  
    vtss\_rx\_master\_timestamp\_get, 676  
    vtss\_rx\_timestamp\_get, 675  
    vtss\_rx\_timestamp\_id\_release, 676  
    vtss\_timestamp\_age, 677  
    vtss\_ts\_adjtimer\_get, 668  
    vtss\_ts\_adjtimer\_one\_sec, 666  
    vtss\_ts\_adjtimer\_set, 668  
    vtss\_ts\_delay\_asymmetry\_get, 673  
    vtss\_ts\_delay\_asymmetry\_set, 673  
    vtss\_ts\_egress\_latency\_get, 672  
    vtss\_ts\_egress\_latency\_set, 672  
    vtss\_ts\_external\_clock\_mode\_get, 669  
    vtss\_ts\_external\_clock\_mode\_set, 669  
    vtss\_ts\_external\_clock\_saved\_get, 670  
    vtss\_ts\_freq\_offset\_get, 669  
    vtss\_ts\_ingress\_latency\_get, 671  
    vtss\_ts\_ingress\_latency\_set, 670  
    vtss\_ts\_internal\_mode\_get, 675  
    vtss\_ts\_internal\_mode\_set, 674  
    vtss\_ts\_ongoing\_adjustment, 667  
    vtss\_ts\_operation\_mode\_get, 674  
    vtss\_ts\_operation\_mode\_set, 673  
    vtss\_ts\_p2p\_delay\_get, 671  
    vtss\_ts\_p2p\_delay\_set, 671  
    vtss\_ts\_status\_change, 677  
    vtss\_ts\_timeofday\_get, 667  
    vtss\_ts\_timeofday\_next\_pps\_get, 667  
    vtss\_ts\_timeofday\_offset\_set, 666  
    vtss\_ts\_timeofday\_set, 665  
    vtss\_ts\_timeofday\_set\_delta, 665  
    vtss\_tx\_timestamp\_idx\_alloc, 677  
    vtss\_tx\_timestamp\_update, 675  
vtss\_ts\_delay\_asymmetry\_get  
    vtss\_ts\_api.h, 673  
vtss\_ts\_delay\_asymmetry\_set  
    vtss\_ts\_api.h, 673  
vtss\_ts\_egress\_latency\_get  
    vtss\_ts\_api.h, 672  
vtss\_ts\_egress\_latency\_set  
    vtss\_ts\_api.h, 672  
vtss\_ts\_ext\_clock\_mode\_t, 274  
    enable, 274  
    freq, 275  
    one\_pps\_mode, 274  
vtss\_ts\_external\_clock\_mode\_get  
    vtss\_ts\_api.h, 669  
vtss\_ts\_external\_clock\_mode\_set  
    vtss\_ts\_api.h, 669  
vtss\_ts\_external\_clock\_saved\_get  
    vtss\_ts\_api.h, 670  
vtss\_ts\_freq\_offset\_get  
    vtss\_ts\_api.h, 669  
vtss\_ts\_id\_t, 275  
    ts\_id, 275  
vtss\_ts\_ingress\_latency\_get  
    vtss\_ts\_api.h, 671  
vtss\_ts\_ingress\_latency\_set  
    vtss\_ts\_api.h, 670  
vtss\_ts\_internal\_mode\_get  
    vtss\_ts\_api.h, 675  
vtss\_ts\_internal\_mode\_set  
    vtss\_ts\_api.h, 674  
vtss\_ts\_internal\_mode\_t, 276  
    int\_fmt, 276  
vtss\_ts\_ongoing\_adjustment  
    vtss\_ts\_api.h, 667  
vtss\_ts\_operation\_mode\_get  
    vtss\_ts\_api.h, 674  
vtss\_ts\_operation\_mode\_set  
    vtss\_ts\_api.h, 673  
vtss\_ts\_operation\_mode\_t, 277  
    mode, 277  
vtss\_ts\_p2p\_delay\_get  
    vtss\_ts\_api.h, 671  
vtss\_ts\_p2p\_delay\_set  
    vtss\_ts\_api.h, 671  
vtss\_ts\_status\_change  
    vtss\_ts\_api.h, 677  
vtss\_ts\_timeofday\_get  
    vtss\_ts\_api.h, 667  
vtss\_ts\_timeofday\_next\_pps\_get  
    vtss\_ts\_api.h, 667  
vtss\_ts\_timeofday\_offset\_set  
    vtss\_ts\_api.h, 666  
vtss\_ts\_timeofday\_set  
    vtss\_ts\_api.h, 665  
vtss\_ts\_timeofday\_set\_delta  
    vtss\_ts\_api.h, 665  
vtss\_ts\_timestamp\_alloc\_t, 277  
    cb, 278  
    context, 278  
    port\_mask, 278  
vtss\_ts\_timestamp\_t, 278  
    context, 279  
    id, 279  
    ts, 279  
    ts\_valid, 279  
vtss\_tx\_timestamp\_idx\_alloc

vtss\_ts\_api.h, 677  
 vtss\_tx\_timestamp\_update  
     vtss\_ts\_api.h, 675  
 vtss\_uc\_flood\_members\_get  
     vtss\_l2\_api.h, 457  
 vtss\_uc\_flood\_members\_set  
     vtss\_l2\_api.h, 457  
 vtss\_vcap\_bit\_t  
     types.h, 381  
 vtss\_vcap\_ip\_t, 280  
     mask, 280  
     value, 280  
 vtss\_vcap\_key\_type\_t  
     types.h, 382  
 vtss\_vcap\_u128\_t, 281  
     mask, 281  
     value, 281  
 vtss\_vcap\_u16\_t, 282  
     mask, 282  
     value, 282  
 vtss\_vcap\_u24\_t, 283  
     mask, 283  
     value, 283  
 vtss\_vcap\_u32\_t, 284  
     mask, 284  
     value, 284  
 vtss\_vcap\_u40\_t, 285  
     mask, 285  
     value, 285  
 vtss\_vcap\_u48\_t, 286  
     mask, 286  
     value, 286  
 vtss\_vcap\_u8\_t, 287  
     mask, 287  
     value, 287  
 vtss\_vcap\_udp\_tcp\_t, 288  
     high, 289  
     in\_range, 288  
     low, 288  
 vtss\_vcap\_vid\_t, 289  
     mask, 290  
     value, 289  
 vtss\_vcap\_vr\_t, 290  
     high, 291  
     low, 291  
     mask, 291  
     r, 292  
     type, 291  
     v, 291  
     value, 291  
     vr, 292  
 vtss\_vcap\_vr\_type\_t  
     types.h, 382  
 vtss\_vce\_action\_t, 292  
     policy\_no, 293  
     vid, 292  
 vtss\_vce\_add  
     vtss\_l2\_api.h, 442  
 vtss\_vce\_del  
     vtss\_l2\_api.h, 442  
 vtss\_vce\_frame\_etype\_t, 293  
     data, 294  
     etype, 293  
 vtss\_vce\_frame\_ipv4\_t, 294  
     dport, 295  
     dscp, 295  
     fragment, 294  
     options, 295  
     proto, 295  
     sip, 295  
 vtss\_vce\_frame\_ipv6\_t, 296  
     dport, 297  
     dscp, 296  
     proto, 296  
     sip, 297  
 vtss\_vce\_frame\_llc\_t, 297  
     data, 298  
 vtss\_vce\_frame\_snap\_t, 298  
     data, 298  
 vtss\_vce\_init  
     vtss\_l2\_api.h, 441  
 vtss\_vce\_key\_t, 299  
     etype, 300  
     frame, 301  
     ipv4, 300  
     ipv6, 301  
     llc, 300  
     mac, 299  
     port\_list, 299  
     snap, 300  
     tag, 299  
     type, 300  
 vtss\_vce\_mac\_t, 301  
     dmac\_bc, 302  
     dmac\_mc, 302  
     smac, 302  
 vtss\_vce\_t, 302  
     action, 303  
     id, 303  
     key, 303  
 vtss\_vce\_tag\_t, 303  
     dei, 304  
     pcp, 304  
     s\_tag, 305  
     tagged, 304  
     vid, 304  
 vtss\_vce\_type\_t  
     vtss\_l2\_api.h, 425  
 vtss\_vcl\_port\_conf\_get  
     vtss\_l2\_api.h, 440  
 vtss\_vcl\_port\_conf\_set  
     vtss\_l2\_api.h, 441  
 vtss\_vcl\_port\_conf\_t, 305  
     dmac\_dip, 305  
 vtss\_vdd\_t  
     types.h, 381

vtss\_vid\_mac\_t, 306  
mac, 306  
vid, 306  
vtss\_vlan\_conf\_get  
    vtss\_l2\_api.h, 436  
vtss\_vlan\_conf\_set  
    vtss\_l2\_api.h, 436  
vtss\_vlan\_conf\_t, 307  
    s\_etype, 307  
vtss\_vlan\_frame\_t  
    types.h, 380  
vtss\_vlan\_port\_conf\_get  
    vtss\_l2\_api.h, 437  
vtss\_vlan\_port\_conf\_set  
    vtss\_l2\_api.h, 437  
vtss\_vlan\_port\_conf\_t, 308  
    frame\_type, 309  
    ingress\_filter, 309  
    port\_type, 308  
    pvid, 308  
    untagged\_vid, 308  
vtss\_vlan\_port\_members\_get  
    vtss\_l2\_api.h, 438  
vtss\_vlan\_port\_members\_set  
    vtss\_l2\_api.h, 438  
vtss\_vlan\_port\_type\_t  
    vtss\_l2\_api.h, 425  
vtss\_vlan\_tag\_t, 309  
    dei, 310  
    pcp, 310  
    tpid, 310  
    vid, 310  
vtss\_vlan\_trans\_group\_add  
    vtss\_l2\_api.h, 442  
vtss\_vlan\_trans\_group\_del  
    vtss\_l2\_api.h, 443  
vtss\_vlan\_trans\_group\_get  
    vtss\_l2\_api.h, 443  
vtss\_vlan\_trans\_group\_to\_port\_get  
    vtss\_l2\_api.h, 444  
vtss\_vlan\_trans\_group\_to\_port\_set  
    vtss\_l2\_api.h, 444  
vtss\_vlan\_trans\_grp2vlan\_conf\_t, 311  
    group\_id, 311  
    trans\_vid, 311  
    vid, 311  
vtss\_vlan\_trans\_port2grp\_conf\_t, 312  
    group\_id, 312  
    ports, 312  
vtss\_vlan\_tx\_tag\_get  
    vtss\_l2\_api.h, 439  
vtss\_vlan\_tx\_tag\_set  
    vtss\_l2\_api.h, 440  
vtss\_vlan\_tx\_tag\_t  
    vtss\_l2\_api.h, 425  
vtss\_vlan\_vid\_conf\_get  
    vtss\_l2\_api.h, 439  
vtss\_vlan\_vid\_conf\_set

    vtss\_l2\_api.h, 439  
    learning, 313  
    mirror, 313  
vtss\_vt\_id\_t  
    vtss\_l2\_api.h, 424  
vtss\_wol\_mac\_addr\_t, 314  
    addr, 314  
vtss\_wol\_passwd\_len\_type\_t  
    vtss\_phy\_api.h, 579  
warm\_start\_enable  
    vtss\_init\_conf\_t, 95  
width  
    vtss\_pi\_conf\_t, 195  
wol\_mac  
    vtss\_phy\_wol\_conf\_t, 193  
wol\_pass  
    vtss\_phy\_wol\_conf\_t, 194  
wol\_passwd\_len  
    vtss\_phy\_wol\_conf\_t, 194  
xaui\_rx\_lane\_flip  
    vtss\_port\_conf\_t, 204  
xaui\_tx\_lane\_flip  
    vtss\_port\_conf\_t, 204  
xtr\_qu  
    vtss\_packet\_rx\_meta\_t, 142  
xtr\_qu\_mask  
    vtss\_packet\_rx\_info\_t, 136