

Vitesse API

Generated by Doxygen 1.8.14

Contents

1	Ethernet Virtual Connections	1
1.1	Port Configuration	1
1.2	Policer Configuration	1
1.3	EVCs	2
1.4	ECEs	2
1.5	EVC Port Statistics	2
2	Layer 2	3
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

3 Security	11
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
4 Data Structure Index	13
4.1 Data Structures	13
5 File Index	21
5.1 File List	21
6 Data Structure Documentation	23
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 power_mode	25
6.1.2.10 exc_col_cont	25
6.1.2.11 adv_dis	26
6.1.2.12 max_tags	26
6.1.2.13 oper_up	26
6.1.2.14 frame_length_chk	26
6.2 serdes_fields_t Struct Reference	26

6.2.1	Detailed Description	27
6.2.2	Field Documentation	27
6.2.2.1	ob_post0	27
6.2.2.2	ob_sr	27
6.3	tag_vtss_fdma_list Struct Reference	27
6.3.1	Detailed Description	28
6.3.2	Field Documentation	28
6.3.2.1	frm_ptr	28
6.3.2.2	act_len	29
6.3.2.3	alloc_ptr	29
6.3.2.4	rx_info	29
6.3.2.5	sw_tstamp	29
6.3.2.6	user	30
6.3.2.7	afi_frm_cnt	30
6.3.2.8	afi_seq_number	30
6.3.2.9	next	30
6.4	vtss_ace_frame_arp_t Struct Reference	31
6.4.1	Detailed Description	31
6.4.2	Field Documentation	31
6.4.2.1	smac	31
6.4.2.2	arp	31
6.4.2.3	req	32
6.4.2.4	unknown	32
6.4.2.5	smac_match	32
6.4.2.6	dmac_match	32
6.4.2.7	length	32
6.4.2.8	ip	33
6.4.2.9	ethernet	33
6.4.2.10	sip	33
6.4.2.11	dip	33

6.5	<code>vtss_ace_frame_etype_t</code> Struct Reference	33
6.5.1	Detailed Description	34
6.5.2	Field Documentation	34
6.5.2.1	<code>dmac</code>	34
6.5.2.2	<code>smac</code>	34
6.5.2.3	<code>etype</code>	34
6.5.2.4	<code>data</code>	35
6.5.2.5	<code>ptp</code>	35
6.6	<code>vtss_ace_frame_ipv4_t</code> Struct Reference	35
6.6.1	Detailed Description	36
6.6.2	Field Documentation	36
6.6.2.1	<code>ttl</code>	36
6.6.2.2	<code>fragment</code>	36
6.6.2.3	<code>options</code>	36
6.6.2.4	<code>ds</code>	36
6.6.2.5	<code>proto</code>	37
6.6.2.6	<code>sip</code>	37
6.6.2.7	<code>dip</code>	37
6.6.2.8	<code>data</code>	37
6.6.2.9	<code>sport</code>	37
6.6.2.10	<code>dport</code>	38
6.6.2.11	<code>tcp_fin</code>	38
6.6.2.12	<code>tcp_syn</code>	38
6.6.2.13	<code>tcp_rst</code>	38
6.6.2.14	<code>tcp_psh</code>	38
6.6.2.15	<code>tcp_ack</code>	39
6.6.2.16	<code>tcp_urg</code>	39
6.6.2.17	<code>sip_eq_dip</code>	39
6.6.2.18	<code>sport_eq_dport</code>	39
6.6.2.19	<code>seq_zero</code>	39

6.6.2.20	ptp	40
6.6.2.21	sip_smac	40
6.7	vtss_ace_frame_ipv6_t Struct Reference	40
6.7.1	Detailed Description	41
6.7.2	Field Documentation	41
6.7.2.1	proto	41
6.7.2.2	sip	41
6.7.2.3	ttl	41
6.7.2.4	ds	41
6.7.2.5	data	42
6.7.2.6	sport	42
6.7.2.7	dport	42
6.7.2.8	tcp_fin	42
6.7.2.9	tcp_syn	42
6.7.2.10	tcp_RST	43
6.7.2.11	tcp_psh	43
6.7.2.12	tcp_ack	43
6.7.2.13	tcp_urg	43
6.7.2.14	sip_eq_dip	43
6.7.2.15	sport_eq_dport	44
6.7.2.16	seq_zero	44
6.7.2.17	ptp	44
6.8	vtss_ace_frame_llc_t Struct Reference	44
6.8.1	Detailed Description	44
6.8.2	Field Documentation	45
6.8.2.1	dmac	45
6.8.2.2	smac	45
6.8.2.3	llc	45
6.9	vtss_ace_frame_snap_t Struct Reference	45
6.9.1	Detailed Description	46

6.9.2 Field Documentation	46
6.9.2.1 dmac	46
6.9.2.2 smac	46
6.9.2.3 snap	46
6.10 vtss_ace_ptp_t Struct Reference	46
6.10.1 Detailed Description	47
6.10.2 Field Documentation	47
6.10.2.1 enable	47
6.10.2.2 header	47
6.11 vtss_ace_sip_smac_t Struct Reference	47
6.11.1 Detailed Description	48
6.11.2 Field Documentation	48
6.11.2.1 enable	48
6.11.2.2 sip	48
6.11.2.3 smac	48
6.12 vtss_ace_t Struct Reference	48
6.12.1 Detailed Description	49
6.12.2 Field Documentation	49
6.12.2.1 id	49
6.12.2.2 port_list	49
6.12.2.3 policy	50
6.12.2.4 type	50
6.12.2.5 action	50
6.12.2.6 dmac_mc	50
6.12.2.7 dmac_bc	50
6.12.2.8 vlan	51
6.12.2.9 etype	51
6.12.2.10 llc	51
6.12.2.11 snap	51
6.12.2.12 arp	51

6.12.2.13 ipv4	52
6.12.2.14 ipv6	52
6.12.2.15 frame	52
6.13 vtss_ace_vlan_t Struct Reference	52
6.13.1 Detailed Description	52
6.13.2 Field Documentation	53
6.13.2.1 vid	53
6.13.2.2 usr_prio	53
6.13.2.3 cfi	53
6.13.2.4 tagged	53
6.14 vtss_acl_action_t Struct Reference	54
6.14.1 Detailed Description	54
6.14.2 Field Documentation	54
6.14.2.1 cpu	54
6.14.2.2 cpu_once	54
6.14.2.3 cpu_queue	55
6.14.2.4 police	55
6.14.2.5 policer_no	55
6.14.2.6 evc_police	55
6.14.2.7 evc_policer_id	55
6.14.2.8 learn	56
6.14.2.9 port_action	56
6.14.2.10 port_list	56
6.14.2.11 mirror	56
6.14.2.12 ptp_action	56
6.15 vtss_acl_policer_conf_t Struct Reference	57
6.15.1 Detailed Description	57
6.15.2 Field Documentation	57
6.15.2.1 bit_rate_enable	57
6.15.2.2 bit_rate	57

6.15.2.3	rate	58
6.16	vtss_acl_port_conf_t Struct Reference	58
6.16.1	Detailed Description	58
6.16.2	Field Documentation	58
6.16.2.1	policy_no	58
6.16.2.2	action	59
6.17	vtss_aggr_mode_t Struct Reference	59
6.17.1	Detailed Description	59
6.17.2	Field Documentation	59
6.17.2.1	smac_enable	59
6.17.2.2	dmac_enable	60
6.17.2.3	sip_dip_enable	60
6.17.2.4	sport_dport_enable	60
6.18	vtss_aneg_t Struct Reference	60
6.18.1	Detailed Description	60
6.18.2	Field Documentation	61
6.18.2.1	obey_pause	61
6.18.2.2	generate_pause	61
6.19	vtss_api_lock_t Struct Reference	61
6.19.1	Detailed Description	61
6.19.2	Field Documentation	62
6.19.2.1	inst	62
6.19.2.2	function	62
6.19.2.3	file	62
6.19.2.4	line	62
6.20	vtss_basic_counters_t Struct Reference	63
6.20.1	Detailed Description	63
6.20.2	Field Documentation	63
6.20.2.1	rx_frames	63
6.20.2.2	tx_frames	63

6.21 vtss_chip_id_t Struct Reference	63
6.21.1 Detailed Description	64
6.21.2 Field Documentation	64
6.21.2.1 part_number	64
6.21.2.2 revision	64
6.22 vtss_counter_pair_t Struct Reference	64
6.22.1 Detailed Description	65
6.22.2 Field Documentation	65
6.22.2.1 frames	65
6.22.2.2 bytes	65
6.23 vtss_debug_info_t Struct Reference	65
6.23.1 Detailed Description	66
6.23.2 Field Documentation	66
6.23.2.1 layer	66
6.23.2.2 group	66
6.23.2.3 chip_no	66
6.23.2.4 port_list	66
6.23.2.5 full	67
6.23.2.6 clear	67
6.23.2.7 vml_format	67
6.24 vtss_debug_lock_t Struct Reference	67
6.24.1 Detailed Description	67
6.24.2 Field Documentation	68
6.24.2.1 chip_no	68
6.25 vtss_dgroup_port_conf_t Struct Reference	68
6.25.1 Detailed Description	68
6.25.2 Field Documentation	68
6.25.2.1 dgroup_no	68
6.26 vtss_dlb_policer_conf_t Struct Reference	69
6.26.1 Detailed Description	69

6.26.2 Field Documentation	69
6.26.2.1 type	69
6.26.2.2 enable	69
6.26.2.3 cf	70
6.26.2.4 line_rate	70
6.26.2.5 cir	70
6.26.2.6 cbs	70
6.26.2.7 eir	70
6.26.2.8 ebs	71
6.27 vtss_ece_action_t Struct Reference	71
6.27.1 Detailed Description	71
6.27.2 Field Documentation	71
6.27.2.1 dir	71
6.27.2.2 pop_tag	72
6.27.2.3 outer_tag	72
6.27.2.4 evc_id	72
6.27.2.5 policy_no	72
6.27.2.6 prio_enable	72
6.27.2.7 prio	73
6.28 vtss_ece_frame_ipv4_t Struct Reference	73
6.28.1 Detailed Description	73
6.28.2 Field Documentation	73
6.28.2.1 dscp	73
6.28.2.2 fragment	74
6.28.2.3 proto	74
6.28.2.4 sip	74
6.28.2.5 sport	74
6.28.2.6 dport	74
6.29 vtss_ece_frame_ipv6_t Struct Reference	75
6.29.1 Detailed Description	75

6.29.2 Field Documentation	75
6.29.2.1 dscp	75
6.29.2.2 proto	75
6.29.2.3 sip	76
6.29.2.4 sport	76
6.29.2.5 dport	76
6.30 vtss_ece_key_t Struct Reference	76
6.30.1 Detailed Description	77
6.30.2 Field Documentation	77
6.30.2.1 port_list	77
6.30.2.2 mac	77
6.30.2.3 tag	77
6.30.2.4 type	77
6.30.2.5 ipv4	78
6.30.2.6 ipv6	78
6.30.2.7 frame	78
6.31 vtss_ece_mac_t Struct Reference	78
6.31.1 Detailed Description	78
6.31.2 Field Documentation	79
6.31.2.1 dmac_mc	79
6.31.2.2 dmac_bc	79
6.31.2.3 smac	79
6.32 vtss_ece_outer_tag_t Struct Reference	79
6.32.1 Detailed Description	80
6.32.2 Field Documentation	80
6.32.2.1 enable	80
6.32.2.2 pcp_dei_preserve	80
6.32.2.3 pcp	80
6.32.2.4 dei	80
6.33 vtss_ece_t Struct Reference	81

6.33.1 Detailed Description	81
6.33.2 Field Documentation	81
6.33.2.1 id	81
6.33.2.2 key	81
6.33.2.3 action	82
6.34 vtss_ece_tag_t Struct Reference	82
6.34.1 Detailed Description	82
6.34.2 Field Documentation	82
6.34.2.1 vid	82
6.34.2.2 pcp	83
6.34.2.3 dei	83
6.34.2.4 tagged	83
6.34.2.5 s_tagged	83
6.35 vtss_eee_port_conf_t Struct Reference	83
6.35.1 Detailed Description	84
6.35.2 Field Documentation	84
6.35.2.1 eee_ena	84
6.35.2.2 eee_fast_queues	84
6.35.2.3 tx_tw	84
6.35.2.4 lp_advertisement	85
6.35.2.5 optimized_for_power	85
6.36 vtss_eee_port_counter_t Struct Reference	85
6.36.1 Detailed Description	85
6.36.2 Field Documentation	85
6.36.2.1 fill_level_get	86
6.36.2.2 fill_level_thres	86
6.36.2.3 fill_level	86
6.36.2.4 tx_out_bytes_get	86
6.36.2.5 tx_out_bytes	86
6.37 vtss_eee_port_state_t Struct Reference	87

6.37.1 Detailed Description	87
6.37.2 Field Documentation	87
6.37.2.1 select	87
6.37.2.2 val	87
6.38 vtss_eps_port_conf_t Struct Reference	87
6.38.1 Detailed Description	88
6.38.2 Field Documentation	88
6.38.2.1 type	88
6.38.2.2 port_no	88
6.39 vtss_evc_conf_t Struct Reference	88
6.39.1 Detailed Description	89
6.39.2 Field Documentation	89
6.39.2.1 learning	89
6.39.2.2 pb	89
6.39.2.3 network	89
6.40 vtss_evc_inner_tag_t Struct Reference	90
6.40.1 Detailed Description	90
6.40.2 Field Documentation	90
6.40.2.1 type	90
6.40.2.2 vid_mode	90
6.40.2.3 vid	91
6.40.2.4 pcp_dei_preserve	91
6.40.2.5 pcp	91
6.40.2.6 dei	91
6.41 vtss_evc_pb_conf_t Struct Reference	91
6.41.1 Detailed Description	92
6.41.2 Field Documentation	92
6.41.2.1 nni	92
6.41.2.2 ivid	92
6.41.2.3 vid	92

6.41.2.4	uvid	93
6.41.2.5	inner_tag	93
6.42	vtss_evc_port_conf_t Struct Reference	93
6.42.1	Detailed Description	93
6.42.2	Field Documentation	93
6.42.2.1	dei_coloring	94
6.42.2.2	inner_tag	94
6.42.2.3	dmac_dip	94
6.43	vtss_fan_conf_t Struct Reference	94
6.43.1	Detailed Description	95
6.43.2	Field Documentation	95
6.43.2.1	fan_pwm_freq	95
6.43.2.2	fan_low_pol	95
6.43.2.3	fan_open_col	95
6.43.2.4	type	95
6.43.2.5	ppr	96
6.44	vtss_fdma_cfg_t Struct Reference	96
6.44.1	Detailed Description	96
6.44.2	Field Documentation	96
6.44.2.1	enable	97
6.44.2.2	rx_mtu	97
6.44.2.3	rx_buf_cnt	97
6.44.2.4	rx_alloc_cb	98
6.44.2.5	rx_dont_strip_vlan_tag	99
6.44.2.6	rx_dont_reinsert_vlan_tag	99
6.44.2.7	rx_allow_vlan_tag_mismatch	99
6.44.2.8	rx_allow_multiple_dcbs	99
6.44.2.9	rx_cb	100
6.44.2.10	tx_buf_cnt	100
6.44.2.11	tx_done_cb	100

6.44.2.12 afi_buf_cnt	101
6.44.2.13 afi_done_cb	101
6.45 vtss_fdma_ch_cfg_t Struct Reference	101
6.45.1 Detailed Description	102
6.45.2 Field Documentation	102
6.45.2.1 usage	102
6.45.2.2 xtr_grp	102
6.45.2.3 inj_grp_mask	103
6.45.2.4 list	103
6.45.2.5 xtr_cb	104
6.45.2.6 prio	104
6.45.2.7 chip_no	105
6.45.2.8 ccm_quotient_max	105
6.46 vtss_fdma_throttle_cfg_t Struct Reference	105
6.46.1 Detailed Description	106
6.46.2 Field Documentation	106
6.46.2.1 frm_limit_per_tick	106
6.46.2.2 byte_limit_per_tick	107
6.46.2.3 suspend_tick_cnt	107
6.47 vtss_fdma_tx_info_t Struct Reference	107
6.47.1 Detailed Description	107
6.47.2 Field Documentation	108
6.47.2.1 pre_cb_ctxt1	108
6.47.2.2 pre_cb_ctxt2	108
6.47.2.3 pre_cb	108
6.47.2.4 afi_fps	109
6.47.2.5 afi_type	109
6.47.2.6 afi_enable_counting	109
6.47.2.7 afi_enable_sequence_numbering	110
6.47.2.8 afi_sequence_number_offset	110

6.48 vtss_init_conf_t Struct Reference	111
6.48.1 Detailed Description	111
6.48.2 Field Documentation	111
6.48.2.1 reg_read	111
6.48.2.2 reg_write	112
6.48.2.3 miim_read	112
6.48.2.4 miim_write	112
6.48.2.5 mmd_read	112
6.48.2.6 mmd_read_inc	112
6.48.2.7 mmd_write	113
6.48.2.8 spi_read_write	113
6.48.2.9 spi_32bit_read_write	113
6.48.2.10 spi_64bit_read_write	113
6.48.2.11 warm_start_enable	113
6.48.2.12 restart_info_src	114
6.48.2.13 restart_info_port	114
6.48.2.14 mux_mode	114
6.48.2.15 pi	114
6.48.2.16 serdes	114
6.49 vtss_inst_create_t Struct Reference	115
6.49.1 Detailed Description	115
6.49.2 Field Documentation	115
6.49.2.1 target	115
6.50 vtss_intr_t Struct Reference	115
6.50.1 Detailed Description	116
6.50.2 Field Documentation	116
6.50.2.1 link_change	116
6.51 vtss_ip_addr_t Struct Reference	116
6.51.1 Detailed Description	116
6.51.2 Field Documentation	116

6.51.2.1 type	117
6.51.2.2 ipv4	117
6.51.2.3 ipv6	117
6.51.2.4 addr	117
6.52 vtss_ip_network_t Struct Reference	117
6.52.1 Detailed Description	118
6.52.2 Field Documentation	118
6.52.2.1 address	118
6.52.2.2 prefix_size	118
6.53 vtss_ipv4_network_t Struct Reference	118
6.53.1 Detailed Description	119
6.53.2 Field Documentation	119
6.53.2.1 address	119
6.53.2.2 prefix_size	119
6.54 vtss_ipv4_uc_t Struct Reference	119
6.54.1 Detailed Description	120
6.54.2 Field Documentation	120
6.54.2.1 network	120
6.54.2.2 destination	120
6.55 vtss_ipv6_network_t Struct Reference	120
6.55.1 Detailed Description	121
6.55.2 Field Documentation	121
6.55.2.1 address	121
6.55.2.2 prefix_size	121
6.56 vtss_ipv6_t Struct Reference	121
6.56.1 Detailed Description	121
6.56.2 Field Documentation	122
6.56.2.1 addr	122
6.57 vtss_ipv6_uc_t Struct Reference	122
6.57.1 Detailed Description	122

6.57.2 Field Documentation	122
6.57.2.1 network	122
6.57.2.2 destination	123
6.58 vtss_irq_conf_t Struct Reference	123
6.58.1 Detailed Description	123
6.58.2 Field Documentation	123
6.58.2.1 external	123
6.58.2.2 destination	124
6.59 vtss_irq_status_t Struct Reference	124
6.59.1 Detailed Description	124
6.59.2 Field Documentation	124
6.59.2.1 active	124
6.59.2.2 raw_ident	125
6.59.2.3 raw_status	125
6.59.2.4 raw_mask	125
6.60 vtss_l3_counters_t Struct Reference	125
6.60.1 Detailed Description	126
6.60.2 Field Documentation	126
6.60.2.1 ipv4uc_received_octets	126
6.60.2.2 ipv4uc_received_frames	126
6.60.2.3 ipv6uc_received_octets	126
6.60.2.4 ipv6uc_received_frames	126
6.60.2.5 ipv4uc_transmitted_octets	127
6.60.2.6 ipv4uc_transmitted_frames	127
6.60.2.7 ipv6uc_transmitted_octets	127
6.60.2.8 ipv6uc_transmitted_frames	127
6.61 vtss_lcpll_status_t Struct Reference	127
6.61.1 Detailed Description	128
6.61.2 Field Documentation	128
6.61.2.1 lock_status	128

6.61.2.2	cal_done	128
6.61.2.3	cal_error	128
6.61.2.4	fsm_lock	129
6.61.2.5	fsm_stat	129
6.61.2.6	gain_stat	129
6.62	vtss_learn_mode_t Struct Reference	129
6.62.1	Detailed Description	129
6.62.2	Field Documentation	130
6.62.2.1	automatic	130
6.62.2.2	cpu	130
6.62.2.3	discard	130
6.63	vtss_mac_t Struct Reference	130
6.63.1	Detailed Description	131
6.63.2	Field Documentation	131
6.63.2.1	addr	131
6.64	vtss_mac_table_entry_t Struct Reference	131
6.64.1	Detailed Description	131
6.64.2	Field Documentation	131
6.64.2.1	vid_mac	132
6.64.2.2	destination	132
6.64.2.3	copy_to_cpu	132
6.64.2.4	locked	132
6.64.2.5	aged	132
6.64.2.6	cpu_queue	133
6.65	vtss_mac_table_status_t Struct Reference	133
6.65.1	Detailed Description	133
6.65.2	Field Documentation	133
6.65.2.1	learned	133
6.65.2.2	replaced	134
6.65.2.3	moved	134

6.65.2.4	aged	134
6.66	vtss_mce_action_t Struct Reference	134
6.66.1	Detailed Description	135
6.66.2	Field Documentation	135
6.66.2.1	policy_no	135
6.66.2.2	prio_enable	135
6.66.2.3	prio	135
6.66.2.4	vid	135
6.66.2.5	pop_cnt	136
6.67	vtss_mce_key_t Struct Reference	136
6.67.1	Detailed Description	136
6.67.2	Field Documentation	136
6.67.2.1	port_list	136
6.67.2.2	vid	137
6.67.2.3	data	137
6.68	vtss_mce_t Struct Reference	137
6.68.1	Detailed Description	137
6.68.2	Field Documentation	137
6.68.2.1	tt_loop	138
6.68.2.2	id	138
6.68.2.3	key	138
6.68.2.4	action	138
6.69	vtss_mirror_conf_t Struct Reference	138
6.69.1	Detailed Description	139
6.69.2	Field Documentation	139
6.69.2.1	port_no	139
6.69.2.2	fwd_enable	139
6.70	vtss_mtimer_t Struct Reference	139
6.70.1	Detailed Description	140
6.70.2	Field Documentation	140

6.70.2.1	timeout	140
6.70.2.2	now	140
6.71	vtss_npi_conf_t Struct Reference	140
6.71.1	Detailed Description	141
6.71.2	Field Documentation	141
6.71.2.1	enable	141
6.71.2.2	port_no	141
6.72	vtss_os_timestamp_t Struct Reference	141
6.72.1	Detailed Description	141
6.72.2	Field Documentation	142
6.72.2.1	hw_cnt	142
6.73	vtss_packet_dma_conf_t Struct Reference	142
6.73.1	Detailed Description	142
6.73.2	Field Documentation	142
6.73.2.1	dma_enable	142
6.74	vtss_packet_frame_info_t Struct Reference	143
6.74.1	Detailed Description	143
6.74.2	Field Documentation	143
6.74.2.1	port_no	143
6.74.2.2	vid	143
6.74.2.3	port_tx	144
6.74.2.4	aggr_rx_disable	144
6.74.2.5	aggr_tx_disable	144
6.75	vtss_packet_port_filter_t Struct Reference	144
6.75.1	Detailed Description	144
6.75.2	Field Documentation	145
6.75.2.1	filter	145
6.75.2.2	tpid	145
6.76	vtss_packet_port_info_t Struct Reference	145
6.76.1	Detailed Description	145

6.76.2 Field Documentation	146
6.76.2.1 port_no	146
6.76.2.2 vid	146
6.76.2.3 aggr_rx_disable	146
6.76.2.4 aggr_tx_disable	146
6.77 vtss_packet_rx_conf_t Struct Reference	147
6.77.1 Detailed Description	147
6.77.2 Field Documentation	147
6.77.2.1 queue	147
6.77.2.2 reg	147
6.77.2.3 map	148
6.77.2.4 grp_map	148
6.78 vtss_packet_rx_header_t Struct Reference	148
6.78.1 Detailed Description	148
6.78.2 Field Documentation	148
6.78.2.1 length	149
6.78.2.2 port_no	149
6.78.2.3 queue_mask	149
6.78.2.4 learn	149
6.78.2.5 arrived_tagged	149
6.78.2.6 tag	150
6.79 vtss_packet_rx_info_t Struct Reference	150
6.79.1 Detailed Description	150
6.79.2 Field Documentation	151
6.79.2.1 hints	151
6.79.2.2 length	151
6.79.2.3 port_no	152
6.79.2.4 glag_no	152
6.79.2.5 tag_type	153
6.79.2.6 tag	153

6.79.2.7	stripped_tag	154
6.79.2.8	xtr_qu_mask	154
6.79.2.9	cos	155
6.79.2.10	acl_hit	155
6.79.2.11	acl_idx	155
6.79.2.12	sw_tstamp	156
6.79.2.13	tstamp_id	156
6.79.2.14	tstamp_id_decoded	156
6.79.2.15	hw_tstamp	157
6.79.2.16	hw_tstamp_decoded	157
6.79.2.17	sflow_type	157
6.79.2.18	sflow_port_no	158
6.79.2.19	oam_info	158
6.79.2.20	oam_info_decoded	158
6.79.2.21	isdx	159
6.80	vtss_packet_rx_meta_t Struct Reference	159
6.80.1	Detailed Description	159
6.80.2	Field Documentation	160
6.80.2.1	no_wait	160
6.80.2.2	chip_no	160
6.80.2.3	xtr_qu	161
6.80.2.4	etype	161
6.80.2.5	fcs	162
6.80.2.6	sw_tstamp	162
6.80.2.7	length	163
6.81	vtss_packet_rx_port_conf_t Struct Reference	163
6.81.1	Detailed Description	163
6.81.2	Field Documentation	164
6.81.2.1	bpdu_reg	164
6.81.2.2	garp_reg	164

6.82 vtss_packet_rx_queue_conf_t Struct Reference	164
6.82.1 Detailed Description	164
6.82.2 Field Documentation	164
6.82.2.1 size	165
6.82.2.2 npi	165
6.83 vtss_packet_rx_queue_map_t Struct Reference	165
6.83.1 Detailed Description	165
6.83.2 Field Documentation	166
6.83.2.1 bpdu_queue	166
6.83.2.2 garp_queue	166
6.83.2.3 learn_queue	166
6.83.2.4 igmp_queue	166
6.83.2.5 ipmc_ctrl_queue	166
6.83.2.6 mac_vid_queue	167
6.83.2.7 stack_queue	167
6.83.2.8 sflow_queue	167
6.83.2.9 lrn_all_queue	167
6.84 vtss_packet_rx_queue_npi_conf_t Struct Reference	167
6.84.1 Detailed Description	168
6.84.2 Field Documentation	168
6.84.2.1 enable	168
6.85 vtss_packet_rx_reg_t Struct Reference	168
6.85.1 Detailed Description	168
6.85.2 Field Documentation	169
6.85.2.1 bpdu_cpu_only	169
6.85.2.2 garp_cpu_only	169
6.85.2.3 ipmc_ctrl_cpu_copy	169
6.85.2.4 igmp_cpu_only	169
6.85.2.5 mld_cpu_only	170
6.86 vtss_packet_tx_ifh_t Struct Reference	170

6.86.1	Detailed Description	170
6.86.2	Field Documentation	170
6.86.2.1	length	170
6.86.2.2	ifh	171
6.87	vtss_packet_tx_info_t Struct Reference	171
6.87.1	Detailed Description	171
6.87.2	Field Documentation	172
6.87.2.1	switch_frm	172
6.87.2.2	dst_port_mask	172
6.87.2.3	frm_len	173
6.87.2.4	tag	173
6.87.2.5	aggr_code	174
6.87.2.6	cos	175
6.87.2.7	ptp_action	175
6.87.2.8	ptp_id	176
6.87.2.9	ptp_timestamp	176
6.87.2.10	latch_timestamp	177
6.87.2.11	oam_type	177
6.87.2.12	isdx	178
6.87.2.13	isdx_dont_use	178
6.87.2.14	dp	179
6.87.2.15	masquerade_port	179
6.87.2.16	pdu_offset	180
6.88	vtss_phy_aneg_t Struct Reference	180
6.88.1	Detailed Description	180
6.88.2	Field Documentation	181
6.88.2.1	speed_10m_hdx	181
6.88.2.2	speed_10m_fdx	181
6.88.2.3	speed_100m_hdx	181
6.88.2.4	speed_100m_fdx	181

6.88.2.5 speed_1g_fdx	181
6.88.2.6 speed_1g_hdx	182
6.88.2.7 symmetric_pause	182
6.88.2.8 asymmetric_pause	182
6.88.2.9 tx_remote_fault	182
6.89 vtss_phy_clock_conf_t Struct Reference	182
6.89.1 Detailed Description	183
6.89.2 Field Documentation	183
6.89.2.1 src	183
6.89.2.2 freq	183
6.89.2.3 squelch	183
6.90 vtss_phy_conf_1g_t Struct Reference	184
6.90.1 Detailed Description	184
6.90.2 Field Documentation	184
6.90.2.1 cfg	184
6.90.2.2 val	184
6.90.2.3 master	185
6.91 vtss_phy_conf_t Struct Reference	185
6.91.1 Detailed Description	185
6.91.2 Field Documentation	185
6.91.2.1 mode	185
6.91.2.2 forced	186
6.91.2.3 aneg	186
6.91.2.4 mdi	186
6.91.2.5 flf	186
6.91.2.6 sigdet	186
6.91.2.7 unidir	187
6.91.2.8 mac_if_pcs	187
6.91.2.9 media_if_pcs	187
6.91.2.10 force_ams_sel	187

6.91.2.11 skip_coma	187
6.92 vtss_phy_eee_conf_t Struct Reference	188
6.92.1 Detailed Description	188
6.92.2 Field Documentation	188
6.92.2.1 eee_mode	188
6.92.2.2 eee_ena_phy	188
6.93 vtss_phy_enhanced_led_control_t Struct Reference	188
6.93.1 Detailed Description	189
6.93.2 Field Documentation	189
6.93.2.1 ser_led_output_1	189
6.93.2.2 ser_led_output_2	189
6.93.2.3 ser_led_frame_rate	189
6.93.2.4 ser_led_select	190
6.94 vtss_phy_forced_t Struct Reference	190
6.94.1 Detailed Description	190
6.94.2 Field Documentation	190
6.94.2.1 speed	190
6.94.2.2 fdx	191
6.95 vtss_phy_led_mode_select_t Struct Reference	191
6.95.1 Detailed Description	191
6.95.2 Field Documentation	191
6.95.2.1 mode	191
6.95.2.2 number	192
6.96 vtss_phy_loopback_t Struct Reference	192
6.96.1 Detailed Description	192
6.96.2 Field Documentation	192
6.96.2.1 far_end_enable	192
6.96.2.2 near_end_enable	193
6.96.2.3 connector_enable	193
6.96.2.4 mac_serdes_input_enable	193

6.96.2.5	mac_serd_des_facility_enable	193
6.96.2.6	mac_serd_des_equipment_enable	193
6.96.2.7	media_serd_des_input_enable	194
6.96.2.8	media_serd_des_facility_enable	194
6.96.2.9	media_serd_des_equipment_enable	194
6.97	vtss_phy_mac_serd_pcs_cntl_t Struct Reference	194
6.97.1	Detailed Description	195
6.97.2	Field Documentation	195
6.97.2.1	disable	195
6.97.2.2	restart	195
6.97.2.3	pd_enable	195
6.97.2.4	aneg_restart	195
6.97.2.5	force_adv_ability	196
6.97.2.6	sgmii_in_pre	196
6.97.2.7	sgmii_out_pre	196
6.97.2.8	serdes_aneg_ena	196
6.97.2.9	serdes_pol_inv_in	196
6.97.2.10	serdes_pol_inv_out	197
6.97.2.11	fast_link_stat_ena	197
6.97.2.12	inhibit_odd_start	197
6.98	vtss_phy_media_serd_pcs_cntl_t Struct Reference	197
6.98.1	Detailed Description	198
6.98.2	Field Documentation	198
6.98.2.1	remote_fault	198
6.98.2.2	aneg_pd_detect	198
6.98.2.3	force_adv_ability	198
6.98.2.4	serdes_pol_inv_in	198
6.98.2.5	serdes_pol_inv_out	199
6.98.2.6	inhibit_odd_start	199
6.98.2.7	force_hls	199

6.98.2.8 force_fefi	199
6.98.2.9 force_fefi_value	199
6.99 vtss_phy_power_conf_t Struct Reference	200
6.99.1 Detailed Description	200
6.99.2 Field Documentation	200
6.99.2.1 mode	200
6.100vtss_phy_power_status_t Struct Reference	200
6.100.1 Detailed Description	201
6.100.2 Field Documentation	201
6.100.2.1 level	201
6.101vtss_phy_reset_conf_t Struct Reference	201
6.101.1 Detailed Description	201
6.101.2 Field Documentation	202
6.101.2.1 mac_if	202
6.101.2.2 media_if	202
6.101.2.3 rgmii	202
6.101.2.4 tbi	202
6.101.2.5 force	202
6.101.2.6 pkt_mode	203
6.101.2.7 i_cpu_en	203
6.102vtss_phy_rgmii_conf_t Struct Reference	203
6.102.1 Detailed Description	203
6.102.2 Field Documentation	203
6.102.2.1 rx_clk_skew_ps	204
6.102.2.2 tx_clk_skew_ps	204
6.103vtss_phy_statistic_t Struct Reference	204
6.103.1 Detailed Description	204
6.103.2 Field Documentation	204
6.103.2.1 cu_good	205
6.103.2.2 cu_bad	205

6.103.2.3 serdes_tx_good	205
6.103.2.4 serdes_tx_bad	205
6.103.2.5 rx_err_cnt_base_tx	205
6.103.2.6 media_mac_serdes_good	206
6.103.2.7 media_mac_serdes_crc	206
6.104vtss_phy_status_1g_t Struct Reference	206
6.104.1 Detailed Description	206
6.104.2 Field Documentation	206
6.104.2.1 master_cfg_fault	207
6.104.2.2 master	207
6.105vtss_phy_tbi_conf_t Struct Reference	207
6.105.1 Detailed Description	207
6.105.2 Field Documentation	207
6.105.2.1 aneg_enable	208
6.106vtss_phy_type_t Struct Reference	208
6.106.1 Detailed Description	208
6.106.2 Field Documentation	208
6.106.2.1 part_number	208
6.106.2.2 revision	209
6.106.2.3 port_cnt	209
6.106.2.4 channel_id	209
6.106.2.5 base_port_no	209
6.106.2.6 phy_api_base_no	209
6.107vtss_phy_veriphy_result_t Struct Reference	210
6.107.1 Detailed Description	210
6.107.2 Field Documentation	210
6.107.2.1 link	210
6.107.2.2 status	210
6.107.2.3 length	211
6.108vtss_phy_wol_conf_t Struct Reference	211

6.108.1 Detailed Description	211
6.108.2 Field Documentation	211
6.108.2.1 secure_on_enable	211
6.108.2.2 wol_mac	212
6.108.2.3 wol_pass	212
6.108.2.4 wol_passwd_len	212
6.108.2.5 magic_pkt_cnt	212
6.109vtss_pi_conf_t Struct Reference	212
6.109.1 Detailed Description	213
6.109.2 Field Documentation	213
6.109.2.1 width	213
6.109.2.2 use_extended_bus_cycle	213
6.109.2.3 cs_wait_ns	213
6.110vtss_policer_ext_t Struct Reference	214
6.110.1 Detailed Description	214
6.110.2 Field Documentation	214
6.110.2.1 frame_rate	214
6.110.2.2 flow_control	214
6.111vtss_policer_t Struct Reference	214
6.111.1 Detailed Description	215
6.111.2 Field Documentation	215
6.111.2.1 level	215
6.111.2.2 rate	215
6.112vtss_port_bridge_counters_t Struct Reference	215
6.112.1 Detailed Description	216
6.112.2 Field Documentation	216
6.112.2.1 dot1dTpPortInDiscards	216
6.113vtss_port_clause_37_adv_t Struct Reference	216
6.113.1 Detailed Description	216
6.113.2 Field Documentation	217

6.113.2.1 <code>fdx</code>	217
6.113.2.2 <code>hdx</code>	217
6.113.2.3 <code>symmetric_pause</code>	217
6.113.2.4 <code>asymmetric_pause</code>	217
6.113.2.5 <code>remote_fault</code>	217
6.113.2.6 <code>acknowledge</code>	218
6.113.2.7 <code>next_page</code>	218
6.114 <code>vtss_port_clause_37_control_t</code> Struct Reference	218
6.114.1 Detailed Description	218
6.114.2 Field Documentation	218
6.114.2.1 <code>enable</code>	219
6.114.2.2 <code>advertisement</code>	219
6.115 <code>vtss_port_conf_t</code> Struct Reference	219
6.115.1 Detailed Description	220
6.115.2 Field Documentation	220
6.115.2.1 <code>if_type</code>	220
6.115.2.2 <code>sd_enable</code>	220
6.115.2.3 <code>sd_active_high</code>	220
6.115.2.4 <code>sd_internal</code>	220
6.115.2.5 <code>frame_gaps</code>	221
6.115.2.6 <code>power_down</code>	221
6.115.2.7 <code>speed</code>	221
6.115.2.8 <code>fdx</code>	221
6.115.2.9 <code>flow_control</code>	221
6.115.2.10 <code>max_frame_length</code>	222
6.115.2.11 <code>frame_length_chk</code>	222
6.115.2.12 <code>max_tags</code>	222
6.115.2.13 <code>exc_col_cont</code>	222
6.115.2.14 <code>xauí_rx_lane_flip</code>	222
6.115.2.15 <code>xauí_tx_lane_flip</code>	223

6.115.2.16oop	223
6.115.2.17serdes	223
6.116vtss_port_counters_t Struct Reference	223
6.116.1 Detailed Description	224
6.116.2 Field Documentation	224
6.116.2.1 rmon	224
6.116.2.2 if_group	224
6.116.2.3 ethernet_like	224
6.116.2.4 bridge	224
6.116.2.5 prop	225
6.116.2.6 evc	225
6.117vtss_port_ethernet_like_counters_t Struct Reference	225
6.117.1 Detailed Description	225
6.117.2 Field Documentation	225
6.117.2.1 dot3InPauseFrames	226
6.117.2.2 dot3OutPauseFrames	226
6.118vtss_port_evc_counters_t Struct Reference	226
6.118.1 Detailed Description	226
6.118.2 Field Documentation	226
6.118.2.1 rx_green	227
6.118.2.2 rx_yellow	227
6.118.2.3 rx_red	227
6.118.2.4 rx_green_discard	227
6.118.2.5 rx_yellow_discard	227
6.118.2.6 tx_green	228
6.118.2.7 tx_yellow	228
6.119vtss_port_flow_control_conf_t Struct Reference	228
6.119.1 Detailed Description	228
6.119.2 Field Documentation	228
6.119.2.1 obey	229

6.119.2.2 generate	229
6.119.2.3 smac	229
6.119.2.4 pfc	229
6.120vtss_port_frame_gaps_t Struct Reference	229
6.120.1 Detailed Description	230
6.120.2 Field Documentation	230
6.120.2.1 hdx_gap_1	230
6.120.2.2 hdx_gap_2	230
6.120.2.3 fdx_gap	230
6.121vtss_port_if_group_counters_t Struct Reference	231
6.121.1 Detailed Description	231
6.121.2 Field Documentation	231
6.121.2.1 ifInOctets	231
6.121.2.2 ifInUcastPkts	231
6.121.2.3 ifInMulticastPkts	232
6.121.2.4 ifInBroadcastPkts	232
6.121.2.5 ifInNUcastPkts	232
6.121.2.6 ifInDiscards	232
6.121.2.7 ifInErrors	232
6.121.2.8 ifOutOctets	233
6.121.2.9 ifOutUcastPkts	233
6.121.2.10fOutMulticastPkts	233
6.121.2.11fOutBroadcastPkts	233
6.121.2.12fOutNUcastPkts	233
6.121.2.13fOutDiscards	234
6.121.2.14fOutErrors	234
6.122vtss_port_map_t Struct Reference	234
6.122.1 Detailed Description	234
6.122.2 Field Documentation	234
6.122.2.1 chip_port	235

6.122.2.2 chip_no	235
6.122.2.3 miim_controller	235
6.122.2.4 miim_addr	235
6.122.2.5 miim_chip_no	235
6.123vtss_port_proprietary_counters_t Struct Reference	236
6.123.1 Detailed Description	236
6.123.2 Field Documentation	236
6.123.2.1 rx_prio	236
6.123.2.2 tx_prio	236
6.124vtss_port_rmon_counters_t Struct Reference	236
6.124.1 Detailed Description	237
6.124.2 Field Documentation	237
6.124.2.1 rx_etherStatsDropEvents	237
6.124.2.2 rx_etherStatsOctets	238
6.124.2.3 rx_etherStatsPkts	238
6.124.2.4 rx_etherStatsBroadcastPkts	238
6.124.2.5 rx_etherStatsMulticastPkts	238
6.124.2.6 rx_etherStatsCRCAlignErrors	238
6.124.2.7 rx_etherStatsUndersizePkts	239
6.124.2.8 rx_etherStatsOversizePkts	239
6.124.2.9 rx_etherStatsFragments	239
6.124.2.10rx_etherStatsJabbers	239
6.124.2.11rx_etherStatsPkts64Octets	239
6.124.2.12rx_etherStatsPkts65to127Octets	240
6.124.2.13rx_etherStatsPkts128to255Octets	240
6.124.2.14rx_etherStatsPkts256to511Octets	240
6.124.2.15rx_etherStatsPkts512to1023Octets	240
6.124.2.16rx_etherStatsPkts1024to1518Octets	240
6.124.2.17rx_etherStatsPkts1519toMaxOctets	241
6.124.2.18x_etherStatsDropEvents	241

6.124.2.19x_etherStatsOctets	241
6.124.2.20x_etherStatsPkts	241
6.124.2.21tx_etherStatsBroadcastPkts	241
6.124.2.22x_etherStatsMulticastPkts	242
6.124.2.23x_etherStatsCollisions	242
6.124.2.24tx_etherStatsPkts64Octets	242
6.124.2.25x_etherStatsPkts65to127Octets	242
6.124.2.26x_etherStatsPkts128to255Octets	242
6.124.2.27x_etherStatsPkts256to511Octets	243
6.124.2.28x_etherStatsPkts512to1023Octets	243
6.124.2.29x_etherStatsPkts1024to1518Octets	243
6.124.2.30x_etherStatsPkts1519toMaxOctets	243
6.125vtss_port_serdes_conf_t Struct Reference	243
6.125.1 Detailed Description	244
6.125.2 Field Documentation	244
6.125.2.1 sfp_dac	244
6.126vtss_port_sgmii_aneg_t Struct Reference	244
6.126.1 Detailed Description	244
6.126.2 Field Documentation	245
6.126.2.1 link	245
6.126.2.2 fdx	245
6.126.2.3 hdx	245
6.126.2.4 speed_10M	245
6.126.2.5 speed_100M	245
6.126.2.6 speed_1G	246
6.126.2.7 aneg_complete	246
6.127vtss_port_status_t Struct Reference	246
6.127.1 Detailed Description	246
6.127.2 Field Documentation	247
6.127.2.1 link_down	247

6.127.2.2 link	247
6.127.2.3 speed	247
6.127.2.4 fdx	247
6.127.2.5 remote_fault	247
6.127.2.6 aneg_complete	248
6.127.2.7 unidirectional_ability	248
6.127.2.8 aneg	248
6.127.2.9 mdi_cross	248
6.127.2.10fiber	248
6.127.2.11copper	249
6.128vtss_qce_action_t Struct Reference	249
6.128.1 Detailed Description	249
6.128.2 Field Documentation	249
6.128.2.1 prio_enable	249
6.128.2.2 prio	250
6.128.2.3 dp_enable	250
6.128.2.4 dp	250
6.128.2.5 dscp_enable	250
6.128.2.6 dscp	250
6.128.2.7 pcp_dei_enable	251
6.128.2.8 pcp	251
6.128.2.9 dei	251
6.128.2.10policy_no_enable	251
6.128.2.11policy_no	251
6.129vtss_qce_frame_etype_t Struct Reference	252
6.129.1 Detailed Description	252
6.129.2 Field Documentation	252
6.129.2.1 etype	252
6.129.2.2 data	252
6.130vtss_qce_frame_ipv4_t Struct Reference	252

6.130.1 Detailed Description	253
6.130.2 Field Documentation	253
6.130.2.1 fragment	253
6.130.2.2 dscp	253
6.130.2.3 proto	253
6.130.2.4 sip	254
6.130.2.5 sport	254
6.130.2.6 dport	254
6.131vtss_qce_frame_ipv6_t Struct Reference	254
6.131.1 Detailed Description	255
6.131.2 Field Documentation	255
6.131.2.1 dscp	255
6.131.2.2 proto	255
6.131.2.3 sip	255
6.131.2.4 sport	255
6.131.2.5 dport	256
6.132vtss_qce_frame_llc_t Struct Reference	256
6.132.1 Detailed Description	256
6.132.2 Field Documentation	256
6.132.2.1 data	256
6.133vtss_qce_frame_snap_t Struct Reference	257
6.133.1 Detailed Description	257
6.133.2 Field Documentation	257
6.133.2.1 data	257
6.134vtss_qce_key_t Struct Reference	257
6.134.1 Detailed Description	258
6.134.2 Field Documentation	258
6.134.2.1 port_list	258
6.134.2.2 mac	258
6.134.2.3 tag	258

6.134.2.4 type	258
6.134.2.5 etype	259
6.134.2.6 llc	259
6.134.2.7 snap	259
6.134.2.8 ipv4	259
6.134.2.9 ipv6	259
6.134.2.10 frame	260
6.135vtss_qce_mac_t Struct Reference	260
6.135.1 Detailed Description	260
6.135.2 Field Documentation	260
6.135.2.1 dmac_mc	260
6.135.2.2 dmac_bc	261
6.135.2.3 smac	261
6.136vtss_qce_t Struct Reference	261
6.136.1 Detailed Description	261
6.136.2 Field Documentation	261
6.136.2.1 id	262
6.136.2.2 key	262
6.136.2.3 action	262
6.137vtss_qce_tag_t Struct Reference	262
6.137.1 Detailed Description	263
6.137.2 Field Documentation	263
6.137.2.1 vid	263
6.137.2.2 pcp	263
6.137.2.3 dei	263
6.137.2.4 tagged	263
6.137.2.5 s_tag	264
6.138vtss_qos_conf_t Struct Reference	264
6.138.1 Detailed Description	264
6.138.2 Field Documentation	264

6.138.2.1 prios	265
6.138.2.2 dscp_trust	265
6.138.2.3 dscp_qos_class_map	265
6.138.2.4 dscp_dp_level_map	265
6.138.2.5 dscp_qos_map	265
6.138.2.6 dscp_qos_map_dp1	266
6.138.2.7 dscp_remark	266
6.138.2.8 dscp_translate_map	266
6.138.2.9 dscp_remap	266
6.138.2.10 dscp_remap_dp1	266
6.138.2.11 policer_uc	267
6.138.2.12 policer_uc_frame_rate	267
6.138.2.13 policer_uc_mode	267
6.138.2.14 policer_mc	267
6.138.2.15 policer_mc_frame_rate	267
6.138.2.16 policer_mc_mode	268
6.138.2.17 policer_bc	268
6.138.2.18 policer_bc_frame_rate	268
6.138.2.19 policer_bc_mode	268
6.139 vtss_qos_port_conf_t Struct Reference	268
6.139.1 Detailed Description	269
6.139.2 Field Documentation	269
6.139.2.1 policer_port	269
6.139.2.2 policer_ext_port	270
6.139.2.3 policer_queue	270
6.139.2.4 shaper_port	270
6.139.2.5 shaper_queue	270
6.139.2.6 excess_enable	270
6.139.2.7 default_prio	271
6.139.2.8 usr_prio	271

6.139.2.9 default_dpl	271
6.139.2.10 default_dei	271
6.139.2.11 tag_class_enable	271
6.139.2.12 qos_class_map	272
6.139.2.13 dp_level_map	272
6.139.2.14 dscp_class_enable	272
6.139.2.15 dscp_mode	272
6.139.2.16 dscp_emode	272
6.139.2.17 dscp_translate	273
6.139.2.18 ag_remark_mode	273
6.139.2.19 tag_default_pcp	273
6.139.2.20 tag_default_dei	273
6.139.2.21 tag_pcp_map	273
6.139.2.22 tag_dei_map	274
6.139.2.23 dwrr_enable	274
6.139.2.24 queue_pct	274
6.139.2.25 dmac_dip	274
6.140 vtss_rcpll_status_t Struct Reference	274
6.140.1 Detailed Description	275
6.140.2 Field Documentation	275
6.140.2.1 out_of_range	275
6.140.2.2 cal_error	275
6.140.2.3 cal_not_done	275
6.141 vtss_restart_status_t Struct Reference	276
6.141.1 Detailed Description	276
6.141.2 Field Documentation	276
6.141.2.1 restart	276
6.141.2.2 prev_version	276
6.141.2.3 cur_version	277
6.142 vtss_routing_entry_t Struct Reference	277

6.142.1 Detailed Description	277
6.142.2 Field Documentation	277
6.142.2.1 type	277
6.142.2.2 ipv4_uc	278
6.142.2.3 ipv6_uc	278
6.142.2.4 route	278
6.142.2.5 vlan	278
6.143vtss_secure_on_passwd_t Struct Reference	278
6.143.1 Detailed Description	279
6.143.2 Field Documentation	279
6.143.2.1 passwd	279
6.144vtss_sedes_macro_conf_t Struct Reference	279
6.144.1 Detailed Description	279
6.144.2 Field Documentation	279
6.144.2.1 serdes1g_vdd	280
6.144.2.2 serdes6g_vdd	280
6.144.2.3 ib_cterm_ena	280
6.144.2.4 qsgmii	280
6.145vtss_sfllow_port_conf_t Struct Reference	280
6.145.1 Detailed Description	281
6.145.2 Field Documentation	281
6.145.2.1 type	281
6.145.2.2 sampling_rate	281
6.146vtss_sgpi_conf_t Struct Reference	281
6.146.1 Detailed Description	282
6.146.2 Field Documentation	282
6.146.2.1 bmode	282
6.146.2.2 bit_count	282
6.146.2.3 port_conf	282
6.147vtss_sgpi_port_conf_t Struct Reference	283

6.147.1 Detailed Description	283
6.147.2 Field Documentation	283
6.147.2.1 enabled	283
6.147.2.2 mode	283
6.147.2.3 int_pol_high	284
6.148vtss_sgpi_port_data_t Struct Reference	284
6.148.1 Detailed Description	284
6.148.2 Field Documentation	284
6.148.2.1 value	284
6.149vtss_shaper_t Struct Reference	285
6.149.1 Detailed Description	285
6.149.2 Field Documentation	285
6.149.2.1 level	285
6.149.2.2 rate	285
6.150vtss_sync_clock_in_t Struct Reference	286
6.150.1 Detailed Description	286
6.150.2 Field Documentation	286
6.150.2.1 port_no	286
6.150.2.2 squelsh	286
6.150.2.3 enable	287
6.151vtss_sync_clock_out_t Struct Reference	287
6.151.1 Detailed Description	287
6.151.2 Field Documentation	287
6.151.2.1 divider	287
6.151.2.2 enable	288
6.152vtss_tci_t Struct Reference	288
6.152.1 Detailed Description	288
6.152.2 Field Documentation	288
6.152.2.1 vid	288
6.152.2.2 cfi	289

6.152.2.3 tagprior	289
6.153vtss_timeofday_t Struct Reference	289
6.153.1 Detailed Description	289
6.153.2 Field Documentation	289
6.153.2.1 sec [1/2]	290
6.153.2.2 sec [2/2]	290
6.154vtss_timestamp_t Struct Reference	290
6.154.1 Detailed Description	290
6.154.2 Field Documentation	290
6.154.2.1 sec_msb	291
6.154.2.2 seconds	291
6.154.2.3 nanoseconds	291
6.155vtss_trace_conf_t Struct Reference	291
6.155.1 Detailed Description	291
6.155.2 Field Documentation	292
6.155.2.1 level	292
6.156vtss_ts_ext_clock_mode_t Struct Reference	292
6.156.1 Detailed Description	292
6.156.2 Field Documentation	292
6.156.2.1 one_pps_mode	292
6.156.2.2 enable	293
6.156.2.3 freq	293
6.157vtss_ts_id_t Struct Reference	293
6.157.1 Detailed Description	293
6.157.2 Field Documentation	293
6.157.2.1 ts_id	294
6.158vtss_ts_internal_mode_t Struct Reference	294
6.158.1 Detailed Description	294
6.158.2 Field Documentation	294
6.158.2.1 int_fmt	294

6.159vtss_ts_operation_mode_t Struct Reference	295
6.159.1 Detailed Description	295
6.159.2 Field Documentation	295
6.159.2.1 mode	295
6.160vtss_ts_timestamp_alloc_t Struct Reference	295
6.160.1 Detailed Description	296
6.160.2 Field Documentation	296
6.160.2.1 port_mask	296
6.160.2.2 context	296
6.160.2.3 cb	296
6.161vtss_ts_timestamp_t Struct Reference	296
6.161.1 Detailed Description	297
6.161.2 Field Documentation	297
6.161.2.1 ts	297
6.161.2.2 id	297
6.161.2.3 context	297
6.161.2.4 ts_valid	298
6.162vtss_vcap_ip_t Struct Reference	298
6.162.1 Detailed Description	298
6.162.2 Field Documentation	298
6.162.2.1 value	298
6.162.2.2 mask	299
6.163vtss_vcap_u128_t Struct Reference	299
6.163.1 Detailed Description	299
6.163.2 Field Documentation	299
6.163.2.1 value	299
6.163.2.2 mask	300
6.164vtss_vcap_u16_t Struct Reference	300
6.164.1 Detailed Description	300
6.164.2 Field Documentation	300

6.164.2.1 value	300
6.164.2.2 mask	301
6.165vtss_vcap_u24_t Struct Reference	301
6.165.1 Detailed Description	301
6.165.2 Field Documentation	301
6.165.2.1 value	301
6.165.2.2 mask	302
6.166vtss_vcap_u32_t Struct Reference	302
6.166.1 Detailed Description	302
6.166.2 Field Documentation	302
6.166.2.1 value	302
6.166.2.2 mask	303
6.167vtss_vcap_u40_t Struct Reference	303
6.167.1 Detailed Description	303
6.167.2 Field Documentation	303
6.167.2.1 value	303
6.167.2.2 mask	304
6.168vtss_vcap_u48_t Struct Reference	304
6.168.1 Detailed Description	304
6.168.2 Field Documentation	304
6.168.2.1 value	304
6.168.2.2 mask	305
6.169vtss_vcap_u8_t Struct Reference	305
6.169.1 Detailed Description	305
6.169.2 Field Documentation	305
6.169.2.1 value	305
6.169.2.2 mask	306
6.170vtss_vcap_udp_tcp_t Struct Reference	306
6.170.1 Detailed Description	306
6.170.2 Field Documentation	306

6.170.2.1 in_range	306
6.170.2.2 low	307
6.170.2.3 high	307
6.171vtss_vcap_vid_t Struct Reference	307
6.171.1 Detailed Description	307
6.171.2 Field Documentation	307
6.171.2.1 value	308
6.171.2.2 mask	308
6.172vtss_vcap_vr_t Struct Reference	308
6.172.1 Detailed Description	308
6.172.2 Field Documentation	309
6.172.2.1 type	309
6.172.2.2 value	309
6.172.2.3 mask	309
6.172.2.4 v	309
6.172.2.5 low	309
6.172.2.6 high	310
6.172.2.7 r	310
6.172.2.8 vr	310
6.173vtss_vce_action_t Struct Reference	310
6.173.1 Detailed Description	310
6.173.2 Field Documentation	310
6.173.2.1 vid	311
6.173.2.2 policy_no	311
6.174vtss_vce_frame_etype_t Struct Reference	311
6.174.1 Detailed Description	311
6.174.2 Field Documentation	311
6.174.2.1 etype	312
6.174.2.2 data	312
6.175vtss_vce_frame_ipv4_t Struct Reference	312

6.175.1 Detailed Description	312
6.175.2 Field Documentation	312
6.175.2.1 fragment	313
6.175.2.2 options	313
6.175.2.3 dscp	313
6.175.2.4 proto	313
6.175.2.5 sip	313
6.175.2.6 dport	314
6.176vtss_vce_frame_ipv6_t Struct Reference	314
6.176.1 Detailed Description	314
6.176.2 Field Documentation	314
6.176.2.1 dscp	314
6.176.2.2 proto	315
6.176.2.3 sip	315
6.176.2.4 dport	315
6.177vtss_vce_frame_llc_t Struct Reference	315
6.177.1 Detailed Description	315
6.177.2 Field Documentation	316
6.177.2.1 data	316
6.178vtss_vce_frame_snap_t Struct Reference	316
6.178.1 Detailed Description	316
6.178.2 Field Documentation	316
6.178.2.1 data	316
6.179vtss_vce_key_t Struct Reference	317
6.179.1 Detailed Description	317
6.179.2 Field Documentation	317
6.179.2.1 port_list	317
6.179.2.2 mac	317
6.179.2.3 tag	318
6.179.2.4 type	318

6.179.2.5 etype	318
6.179.2.6 llc	318
6.179.2.7 snap	318
6.179.2.8 ipv4	319
6.179.2.9 ipv6	319
6.179.2.10 frame	319
6.180vtss_vce_mac_t Struct Reference	319
6.180.1 Detailed Description	319
6.180.2 Field Documentation	320
6.180.2.1 dmac_mc	320
6.180.2.2 dmac_bc	320
6.180.2.3 smac	320
6.181vtss_vce_t Struct Reference	320
6.181.1 Detailed Description	321
6.181.2 Field Documentation	321
6.181.2.1 id	321
6.181.2.2 key	321
6.181.2.3 action	321
6.182vtss_vce_tag_t Struct Reference	321
6.182.1 Detailed Description	322
6.182.2 Field Documentation	322
6.182.2.1 vid	322
6.182.2.2 pcp	322
6.182.2.3 dei	322
6.182.2.4 tagged	323
6.182.2.5 s_tag	323
6.183vtss_vcl_port_conf_t Struct Reference	323
6.183.1 Detailed Description	323
6.183.2 Field Documentation	323
6.183.2.1 dmac_dip	324

6.184vtss_vid_mac_t Struct Reference	324
6.184.1 Detailed Description	324
6.184.2 Field Documentation	324
6.184.2.1 vid	324
6.184.2.2 mac	325
6.185vtss_vlan_conf_t Struct Reference	325
6.185.1 Detailed Description	325
6.185.2 Field Documentation	325
6.185.2.1 s_etype	325
6.186vtss_vlan_port_conf_t Struct Reference	326
6.186.1 Detailed Description	326
6.186.2 Field Documentation	326
6.186.2.1 port_type	326
6.186.2.2 pvid	326
6.186.2.3 untagged_vid	327
6.186.2.4 frame_type	327
6.186.2.5 ingress_filter	327
6.187vtss_vlan_tag_t Struct Reference	327
6.187.1 Detailed Description	327
6.187.2 Field Documentation	328
6.187.2.1 tpid	328
6.187.2.2 pcp	328
6.187.2.3 dei	328
6.187.2.4 vid	328
6.188vtss_vlan_trans_grp2vlan_conf_t Struct Reference	329
6.188.1 Detailed Description	329
6.188.2 Field Documentation	329
6.188.2.1 group_id	329
6.188.2.2 vid	329
6.188.2.3 trans_vid	330

6.189vtss_vlan_trans_port2grp_conf_t Struct Reference	330
6.189.1 Detailed Description	330
6.189.2 Field Documentation	330
6.189.2.1 group_id	330
6.189.2.2 ports	331
6.190vtss_vlan_vid_conf_t Struct Reference	331
6.190.1 Detailed Description	331
6.190.2 Field Documentation	331
6.190.2.1 learning	331
6.190.2.2 mirror	332
6.191vtss_wol_mac_addr_t Struct Reference	332
6.191.1 Detailed Description	332
6.191.2 Field Documentation	332
6.191.2.1 addr	332
7 File Documentation	333
7.1 vtss_api/include/vtss/api/l2_types.h File Reference	333
7.1.1 Detailed Description	333
7.1.2 Enumeration Type Documentation	333
7.1.2.1 vtss_sflow_type_t	333
7.2 vtss_api/include/vtss/api/options.h File Reference	334
7.2.1 Detailed Description	335
7.2.2 Macro Definition Documentation	335
7.2.2.1 VTSS_ARCH_CARACAL	336
7.2.2.2 VTSS_FEATURE_EVC	336
7.2.2.3 VTSS_FEATURE_QOS_POLICER_DLB	336
7.2.2.4 VTSS_ARCH_LUTON26	336
7.2.2.5 VTSS_FEATURE_MISC	336
7.2.2.6 VTSS_FEATURE_SERIAL_GPIO	337
7.2.2.7 VTSS_FEATURE_PORT_CONTROL	337
7.2.2.8 VTSS_FEATURE_CLAUSE_37	337

7.2.2.9	VTSS_FEATURE_EXC_COL_CONT	337
7.2.2.10	VTSS_FEATURE_PORT_CNT_BRIDGE	337
7.2.2.11	VTSS_FEATURE_QOS	338
7.2.2.12	VTSS_FEATURE_QCL	338
7.2.2.13	VTSS_FEATURE_QCL_V2	338
7.2.2.14	VTSS_FEATURE_QCL_DMAC_DIP	338
7.2.2.15	VTSS_FEATURE_QCL_KEY_S_TAG	338
7.2.2.16	VTSS_FEATURE_QCL_PCP_DEI_ACTION	339
7.2.2.17	VTSS_FEATURE_QCL_POLICY_ACTION	339
7.2.2.18	VTSS_FEATURE_QOS_POLICER_UC_SWITCH	339
7.2.2.19	VTSS_FEATURE_QOS_POLICER_MC_SWITCH	339
7.2.2.20	VTSS_FEATURE_QOS_POLICER_BC_SWITCH	339
7.2.2.21	VTSS_FEATURE_QOS_PORT_POLICER_EXT	340
7.2.2.22	VTSS_FEATURE_QOS_PORT_POLICER_EXT_FPS	340
7.2.2.23	VTSS_FEATURE_QOS_PORT_POLICER_EXT_FC	340
7.2.2.24	VTSS_FEATURE_QOS_QUEUE_POLICER	340
7.2.2.25	VTSS_FEATURE_QOS_QUEUE_TX	340
7.2.2.26	VTSS_FEATURE_QOS_SCHEDULER_V2	341
7.2.2.27	VTSS_FEATURE_QOS_TAG_REMARK_V2	341
7.2.2.28	VTSS_FEATURE_QOS_CLASSIFICATION_V2	341
7.2.2.29	VTSS_FEATURE_QOS_EGRESS_QUEUE_SHAPERS	341
7.2.2.30	VTSS_FEATURE_QOS_EGRESS_QUEUE_SHAPERS_EB	341
7.2.2.31	VTSS_FEATURE_QOS_DSCP_CLASS_DP_AWARE	342
7.2.2.32	VTSS_FEATURE_QOS_DSCP_REMARK	342
7.2.2.33	VTSS_FEATURE_QOS_DSCP_REMARK_V2	342
7.2.2.34	VTSS_FEATURE_QOS_DSCP_REMARK_DP_AWARE	342
7.2.2.35	VTSS_FEATURE_PACKET	342
7.2.2.36	VTSS_FEATURE_PACKET_TX	343
7.2.2.37	VTSS_FEATURE_PACKET_RX	343
7.2.2.38	VTSS_FEATURE_PACKET_GROUPING	343

7.2.2.39	VTSS_FEATURE_PACKET_PORT_REG	343
7.2.2.40	VTSS_FEATURE_LAYER2	343
7.2.2.41	VTSS_FEATURE_PVLAN	344
7.2.2.42	VTSS_FEATURE_VLAN_PORT_V2	344
7.2.2.43	VTSS_FEATURE_VLAN_TX_TAG	344
7.2.2.44	VTSS_FEATURE_IPV4_MC_SIP	344
7.2.2.45	VTSS_FEATURE_IPV6_MC_SIP	344
7.2.2.46	VTSS_FEATURE_MAC_AGE_AUTO	345
7.2.2.47	VTSS_FEATURE_MAC_CPU_QUEUE	345
7.2.2.48	VTSS_FEATURE_EEE	345
7.2.2.49	VTSS_FEATURE_PORT_MUX	345
7.2.2.50	VTSS_FEATURE_FAN	345
7.2.2.51	VTSS_FEATURE_VCAP [1/2]	346
7.2.2.52	VTSS_FEATURE_ACL	346
7.2.2.53	VTSS_FEATURE_ACL_V2	346
7.2.2.54	VTSS_FEATURE_VCL	346
7.2.2.55	VTSS_FEATURE_TIMESTAMP	346
7.2.2.56	VTSS_FEATURE_TIMESTAMP_ONE_STEP	347
7.2.2.57	VTSS_FEATURE_TIMESTAMP_LATENCY_COMP	347
7.2.2.58	VTSS_FEATURE_SYNC	347
7.2.2.59	VTSS_FEATURE_NPI	347
7.2.2.60	VTSS_FEATURE_IRQ_CONTROL	347
7.2.2.61	VTSS_OPT_VCORE_III	348
7.2.2.62	VTSS_FEATURE_FDMA	348
7.2.2.63	VTSS_FEATURE_AFI_FDMA	348
7.2.2.64	VTSS_FEATURE_LED_POW_REDUC	348
7.2.2.65	VTSS_FEATURE_INTERRUPTS	348
7.2.2.66	VTSS_FEATURE_VLAN_TRANSLATION	349
7.2.2.67	VTSS_FEATURE_SFLOW	349
7.2.2.68	VTSS_FEATURE_MIRROR_CPU	349

7.2.2.69	VTSS_FEATURE_SERDES_MACRO_SETTINGS	349
7.2.2.70	VTSS_CHIP CU PHY	349
7.2.2.71	VTSS_OPT_TRACE	350
7.2.2.72	VTSS_OPT_FDMA_IRQ_CONTEXT	350
7.2.2.73	VTSS_OPT_FDMA_DEBUG	350
7.2.2.74	VTSS_OPT_VAUI_EQ_CTRL	350
7.2.2.75	VTSS_OPT_PORT_COUNT	350
7.2.2.76	VTSS_PHY_OPT_VERIPHY	351
7.2.2.77	VTSS_FEATURE_WARM_START	351
7.2.2.78	VTSS_FEATURE_VCAP [2/2]	351
7.3	vtss_api/include/vtss/api/phy.h File Reference	351
7.3.1	Detailed Description	352
7.3.2	Macro Definition Documentation	352
7.3.2.1	VTSS_PHY_POWER_ACTIPHYS_BIT	352
7.3.2.2	VTSS_PHY_POWER_DYNAMIC_BIT	352
7.3.3	Enumeration Type Documentation	352
7.3.3.1	vtss_phy_power_mode_t	352
7.3.3.2	vtss_phy_veriphy_status_t	353
7.4	vtss_api/include/vtss/api/port.h File Reference	353
7.4.1	Detailed Description	355
7.4.2	Macro Definition Documentation	355
7.4.2.1	PORT_CAP_NONE	355
7.4.2.2	PORT_CAP_AUTONEG	356
7.4.2.3	PORT_CAP_10M_HDX	356
7.4.2.4	PORT_CAP_10M_FDX	356
7.4.2.5	PORT_CAP_100M_HDX	356
7.4.2.6	PORT_CAP_100M_FDX	356
7.4.2.7	PORT_CAP_1G_FDX	357
7.4.2.8	PORT_CAP_2_5G_FDX	357
7.4.2.9	PORT_CAP_5G_FDX	357

7.4.2.10 PORT_CAP_10G_FDX	357
7.4.2.11 PORT_CAP_FLOW_CTRL	357
7.4.2.12 PORT_CAP_COPPER	358
7.4.2.13 PORT_CAP_FIBER	358
7.4.2.14 PORT_CAP_DUAL_COPPER	358
7.4.2.15 PORT_CAP_DUAL_FIBER	358
7.4.2.16 PORT_CAP_SD_ENABLE	358
7.4.2.17 PORT_CAP_SD_HIGH	359
7.4.2.18 PORT_CAP_SD_INTERNAL	359
7.4.2.19 PORT_CAP_DUAL_FIBER_100FX	359
7.4.2.20 PORT_CAP_XAUI_LANE_FLIP	359
7.4.2.21 PORT_CAP_VTSS_10G_PHY	359
7.4.2.22 PORT_CAP_SFP_DETECT	360
7.4.2.23 PORT_CAP_STACKING	360
7.4.2.24 PORT_CAP_DUAL_SFP_DETECT	360
7.4.2.25 PORT_CAP_SFP_ONLY	360
7.4.2.26 PORT_CAP_DUAL_COPPER_100FX	360
7.4.2.27 PORT_CAP_HDX	361
7.4.2.28 PORT_CAP_TRI_SPEED_FDX	361
7.4.2.29 PORT_CAP_TRI_SPEED	361
7.4.2.30 PORT_CAP_1G_PHY	361
7.4.2.31 PORT_CAP_TRI_SPEED_COPPER	361
7.4.2.32 PORT_CAP_TRI_SPEED_FIBER	362
7.4.2.33 PORT_CAP_TRI_SPEED_DUAL_COPPER	362
7.4.2.34 PORT_CAP_TRI_SPEED_DUAL_FIBER	362
7.4.2.35 PORT_CAP_TRI_SPEED_DUAL_FIBER_100FX	362
7.4.2.36 PORT_CAP_ANY_FIBER	362
7.4.2.37 PORT_CAP_SPEED_DUAL_ANY_FIBER_FIXED_SPEED	363
7.4.2.38 PORT_CAP_SPEED_DUAL_ANY_FIBER	363
7.4.2.39 PORT_CAP_TRI_SPEED_DUAL_ANY_FIBER	363

7.4.2.40	PORT_CAP_TRI_SPEED_DUAL_ANY_FIBER_FIXED_SFP_SPEED	363
7.4.2.41	PORT_CAP_DUAL_FIBER_1000X	363
7.4.2.42	PORT_CAP_SFP_1G	364
7.4.2.43	PORT_CAP_SFP_2_5G	364
7.4.2.44	PORT_CAP_SFP_SD_HIGH	364
7.4.2.45	PORT_CAP_2_5G_TRI_SPEED_FDX	364
7.4.2.46	PORT_CAP_2_5G_TRI_SPEED	364
7.4.2.47	PORT_CAP_2_5G_TRI_SPEED_COPPER	365
7.4.3	Typedef Documentation	365
7.4.3.1	port_cap_t	365
7.4.4	Enumeration Type Documentation	365
7.4.4.1	vtss_port_speed_t	365
7.4.4.2	vtss_fiber_port_speed_t	366
7.5	vtss_api/include/vtss/api/types.h File Reference	366
7.5.1	Detailed Description	373
7.5.2	Macro Definition Documentation	373
7.5.2.1	PRIu64	373
7.5.2.2	PRIi64	373
7.5.2.3	PRIx64	374
7.5.2.4	VTSS_BIT64	374
7.5.2.5	VTSS_BITMASK64	374
7.5.2.6	VTSS_EXTRACT_BITFIELD64	374
7.5.2.7	VTSS_ENCODE_BITFIELD64	374
7.5.2.8	VTSS_ENCODE_BITMASK64	375
7.5.2.9	TRUE	375
7.5.2.10	FALSE	375
7.5.2.11	VTSS_PACKET_RATE_DISABLED	375
7.5.2.12	VTSS_PORT_COUNT [1/2]	375
7.5.2.13	VTSS_PORT_COUNT [2/2]	376
7.5.2.14	VTSS_PORTS	376

7.5.2.15	VTSS_PORT_NO_NONE	376
7.5.2.16	VTSS_PORT_NO_CPU	376
7.5.2.17	VTSS_PORT_NO_START	376
7.5.2.18	VTSS_PORT_NO_END	377
7.5.2.19	VTSS_PORT_ARRAY_SIZE	377
7.5.2.20	VTSS_PORT_IS_PORT	377
7.5.2.21	VTSS_PRIOS	377
7.5.2.22	VTSS_PRIO_NO_NONE	377
7.5.2.23	VTSS_PRIO_START	378
7.5.2.24	VTSS_PRIO_END	378
7.5.2.25	VTSS_PRIO_ARRAY_SIZE	378
7.5.2.26	VTSS_QUEUES	378
7.5.2.27	VTSS_QUEUE_START	378
7.5.2.28	VTSS_QUEUE_END	379
7.5.2.29	VTSS_QUEUE_ARRAY_SIZE	379
7.5.2.30	VTSS_PCPS	379
7.5.2.31	VTSS_PCP_START	379
7.5.2.32	VTSS_PCP_END	379
7.5.2.33	VTSS_PCP_ARRAY_SIZE	380
7.5.2.34	VTSS_DEIS	380
7.5.2.35	VTSS_DEI_START	380
7.5.2.36	VTSS_DEI_END	380
7.5.2.37	VTSS_DEI_ARRAY_SIZE	380
7.5.2.38	VTSS_DPLS	381
7.5.2.39	VTSS_DPL_START	381
7.5.2.40	VTSS_DPL_END	381
7.5.2.41	VTSS_DPL_ARRAY_SIZE	381
7.5.2.42	VTSS_BITRATE_DISABLED	381
7.5.2.43	VTSS_VID_NULL	382
7.5.2.44	VTSS_VID_DEFAULT	382

7.5.2.45	VTSS_VID_RESERVED	382
7.5.2.46	VTSS_VIDS	382
7.5.2.47	VTSS_VID_ALL	382
7.5.2.48	VTSSETYPE_VTSS	383
7.5.2.49	VTSS_MAC_ADDR_SZ_BYTES	383
7.5.2.50	MAC_ADDR_BROADCAST	383
7.5.2.51	VTSS_EVCS	383
7.5.2.52	VTSS_ISDX_NONE	383
7.5.2.53	VTSS_AGGRS	384
7.5.2.54	VTSS_AGGR_NO_NONE	384
7.5.2.55	VTSS_AGGR_NO_START	384
7.5.2.56	VTSS_AGGR_NO_END	384
7.5.2.57	VTSS_GLAGS	384
7.5.2.58	VTSS_GLAG_NO_NONE	385
7.5.2.59	VTSS_GLAG_NO_START	385
7.5.2.60	VTSS_GLAG_NO_END	385
7.5.2.61	VTSS_GLAG_PORTS	385
7.5.2.62	VTSS_GLAG_PORT_START	385
7.5.2.63	VTSS_GLAG_PORT_END	386
7.5.2.64	VTSS_GLAG_PORT_ARRAY_SIZE	386
7.5.2.65	VTSS_PACKET_RX_QUEUE_CNT	386
7.5.2.66	VTSS_PACKET_RX_GRP_CNT	386
7.5.2.67	VTSS_PACKET_TX_GRP_CNT	386
7.5.2.68	VTSS_PACKET_RX_QUEUE_NONE	387
7.5.2.69	VTSS_PACKET_RX_QUEUE_START	387
7.5.2.70	VTSS_PACKET_RX_QUEUE_END	387
7.5.2.71	VTSS_ACL_POLICERS	387
7.5.2.72	VTSS_ACL_POLICER_NO_START	387
7.5.2.73	VTSS_ACL_POLICER_NO_END	388
7.5.2.74	VTSS_ACL_POLICY_NO_NONE	388

7.5.2.75	VTSS_ACL_POLICY_NO_MIN	388
7.5.2.76	VTSS_ACL_POLICY_NO_MAX	388
7.5.2.77	VTSS_ACL_POLICIES	388
7.5.2.78	VTSS_ACL_POLICY_NO_START	389
7.5.2.79	VTSS_ACL_POLICY_NO_END	389
7.5.2.80	VTSS_HQOS_COUNT	389
7.5.2.81	VTSS_HQOS_ID_NONE	389
7.5.2.82	VTSS_ONE_MIA	389
7.5.2.83	VTSS_ONE_MILL	390
7.5.2.84	VTSS_MAX_TIMEINTERVAL	390
7.5.2.85	VTSS_INTERVAL_SEC	390
7.5.2.86	VTSS_INTERVAL_MS	390
7.5.2.87	VTSS_INTERVAL_US	390
7.5.2.88	VTSS_INTERVAL_NS	391
7.5.2.89	VTSS_INTERVAL_PS	391
7.5.2.90	VTSS_SEC_NS_INTERVAL	391
7.5.2.91	VTSS_CLOCK_IDENTITY_LENGTH	391
7.5.2.92	VTSS_SYNCE_CLK_PORT_ARRAY_SIZE	391
7.5.3	Typedef Documentation	392
7.5.3.1	i8	392
7.5.3.2	i16	392
7.5.3.3	i32	392
7.5.3.4	i64	392
7.5.3.5	u8	393
7.5.3.6	u16	393
7.5.3.7	u32	393
7.5.3.8	u64	393
7.5.3.9	BOOL	393
7.5.3.10	uintptr_t	394
7.5.3.11	vtss_mac_addr_t	394

7.5.3.12 <code>vtss_isdx_t</code>	394
7.5.3.13 <code>vtss_packet_rx_grp_t</code>	394
7.5.3.14 <code>vtss_packet_tx_grp_t</code>	394
7.5.4 Enumeration Type Documentation	394
7.5.4.1 anonymous enum	394
7.5.4.2 <code>vtss_mem_flags_t</code>	397
7.5.4.3 <code>vtss_port_interface_t</code>	397
7.5.4.4 <code>vtss_serdes_mode_t</code>	398
7.5.4.5 <code>vtss_storm_policer_mode_t</code>	399
7.5.4.6 <code>vtss_policer_type_t</code>	399
7.5.4.7 <code>vtss_vlan_frame_t</code>	399
7.5.4.8 <code>vtss_packet_reg_type_t</code>	400
7.5.4.9 <code>vtss_vdd_t</code>	400
7.5.4.10 <code>vtss_ip_type_t</code>	400
7.5.4.11 <code>vtss_vcap_bit_t</code>	401
7.5.4.12 <code>vtss_vcap_vr_type_t</code>	401
7.5.4.13 <code>vtss_vcap_key_type_t</code>	401
7.5.4.14 <code>vtss_ece_dir_t</code>	402
7.5.4.15 <code>vtss_ece_pop_tag_t</code>	402
7.5.4.16 <code>vtss_hqos_sch_mode_t</code>	402
7.6 <code>vtss_api/include/vtss_ae_api.h</code> File Reference	403
7.6.1 Detailed Description	403
7.7 <code>vtss_api/include/vtss_afi_api.h</code> File Reference	403
7.7.1 Detailed Description	403
7.8 <code>vtss_api/include/vtss_aneg_api.h</code> File Reference	403
7.8.1 Detailed Description	403
7.9 <code>vtss_api/include/vtss_api.h</code> File Reference	404
7.9.1 Detailed Description	404
7.10 <code>vtss_api/include/vtss_evc_api.h</code> File Reference	404
7.10.1 Detailed Description	406

7.10.2 Macro Definition Documentation	406
7.10.2.1 VTSS_EVC_POLICERS	406
7.10.2.2 VTSS_EVC_ID_NONE	407
7.10.2.3 VTSS_ECE_ID_LAST	407
7.10.2.4 VTSS_MCE_ID_LAST	407
7.10.2.5 VTSS_MCE_POP_NONE	407
7.10.3 Enumeration Type Documentation	407
7.10.3.1 vtss_evc_vid_mode_t	407
7.10.3.2 vtss_evc_inner_tag_type_t	408
7.10.3.3 vtss_ece_type_t	408
7.10.3.4 vtss_ece_port_t	408
7.10.4 Function Documentation	409
7.10.4.1 vtss_evc_port_conf_get()	409
7.10.4.2 vtss_evc_port_conf_set()	409
7.10.4.3 vtss_evc_policer_conf_get()	410
7.10.4.4 vtss_evc_policer_conf_set()	410
7.10.4.5 vtss_evc_add()	411
7.10.4.6 vtss_evc_del()	411
7.10.4.7 vtss_evc_get()	411
7.10.4.8 vtss_ece_init()	412
7.10.4.9 vtss_ece_add()	412
7.10.4.10 vtss_ece_del()	413
7.10.4.11 vtss_mce_init()	413
7.10.4.12 vtss_mce_add()	413
7.10.4.13 vtss_mce_del()	414
7.11 vtss_api/include/vtss_fdma_api.h File Reference	414
7.11.1 Detailed Description	416
7.11.2 Macro Definition Documentation	416
7.11.2.1 VTSS_FDMA_CH_CNT	416
7.11.2.2 VTSS_PHYS_PORT_CNT	417

7.11.2.3 VTSS_FDMA_DCB_SIZE_BYTES	417
7.11.2.4 VTSS_FDMA_HDR_SIZE_BYTES	417
7.11.2.5 VTSS_FDMA_MAX_DATA_PER_DCB_BYTES	417
7.11.2.6 VTSS_FDMA_MIN_FRAME_SIZE_BYTES	417
7.11.2.7 VTSS_FDMA_MIN_DATA_PER_INJ_SOF_DCB_BYTES	418
7.11.2.8 VTSS_FDMA_MIN_DATA_PER_XTR_SOF_DCB_BYTES	418
7.11.2.9 VTSS_FDMA_MIN_DATA_PER_NON_SOF_DCB_BYTES	418
7.11.2.10 VTSS_FDMA_MAX_FRAME_SIZE_BYTES	418
7.11.2.11 VTSS_OS_DCACHE_LINE_SIZE_BYTES	418
7.11.2.12 VTSS_OS_COMPILER_ATTRIBUTE_ALIGNED	419
7.11.2.13 VTSS_AFI_FPS_MAX	419
7.11.2.14 VTSS_FDMA_CCM_QUOTIENT_MAX	419
7.11.2.15 VTSS_FDMA_CCM_FREQ_LIST_LEN	419
7.11.2.16 VTSS_FDMA_CCM_FPS_MAX	420
7.11.3 Typedef Documentation	420
7.11.3.1 vtss_fdma_ch_t	420
7.11.3.2 vtss_fdma_list_t	422
7.11.4 Enumeration Type Documentation	422
7.11.4.1 vtss_fdma_afi_type_t	422
7.11.4.2 vtss_fdma_ch_usage_t	423
7.11.4.3 vtss_fdma_dcb_type_t	423
7.11.5 Function Documentation	423
7.11.5.1 vtss_fdma_uninit()	424
7.11.5.2 vtss_fdma_cfg()	424
7.11.5.3 vtss_fdma_dcb_release()	425
7.11.5.4 vtss_fdma_tx()	425
7.11.5.5 vtss_fdma_tx_info_init()	426
7.11.5.6 vtss_fdma_afi_cancel()	427
7.11.5.7 vtss_fdma_afi_frm_cnt()	427
7.11.5.8 vtss_fdma_dcb_get()	428

7.11.5.9 <code>vtss_fdma_throttle_cfg_get()</code>	429
7.11.5.10 <code>vtss_fdma_throttle_cfg_set()</code>	429
7.11.5.11 <code>vtss_fdma_throttle_tick()</code>	430
7.11.5.12 <code>vtss_fdma_stats_clr()</code>	430
7.11.5.13 <code>vtss_fdma_irq_handler()</code>	431
7.12 <code>vtss_api/include/vtss_gfp_api.h</code> File Reference	432
7.12.1 Detailed Description	432
7.13 <code>vtss_api/include/vtss_hqos_api.h</code> File Reference	432
7.13.1 Detailed Description	432
7.14 <code>vtss_api/include/vtss_i2c_api.h</code> File Reference	432
7.14.1 Detailed Description	433
7.15 <code>vtss_api/include/vtss_init_api.h</code> File Reference	433
7.15.1 Detailed Description	435
7.15.2 Macro Definition Documentation	435
7.15.2.1 <code>VTSS_I2C_NO_MULTIPLEXER</code>	435
7.15.3 Typedef Documentation	435
7.15.3.1 <code>vtss_reg_read_t</code>	435
7.15.3.2 <code>vtss_reg_write_t</code>	436
7.15.3.3 <code>vtss_i2c_read_t</code>	436
7.15.3.4 <code>vtss_i2c_write_t</code>	437
7.15.3.5 <code>vtss_spi_read_write_t</code>	437
7.15.3.6 <code>vtss_spi_32bit_read_write_t</code>	438
7.15.3.7 <code>vtss_spi_64bit_read_write_t</code>	438
7.15.3.8 <code>vtss_miim_read_t</code>	439
7.15.3.9 <code>vtss_miim_write_t</code>	439
7.15.3.10 <code>vtss_mmd_read_t</code>	440
7.15.3.11 <code>vtss_mmd_read_inc_t</code>	440
7.15.3.12 <code>vtss_mmd_write_t</code>	441
7.15.4 Enumeration Type Documentation	441
7.15.4.1 <code>vtss_target_type_t</code>	441

7.15.4.2 vtss_port_mux_mode_t	442
7.15.4.3 vtss_restart_t	442
7.15.5 Function Documentation	443
7.15.5.1 vtss_inst_get()	443
7.15.5.2 vtss_inst_create()	443
7.15.5.3 vtss_inst_destroy()	444
7.15.5.4 vtss_init_conf_get()	444
7.15.5.5 vtss_init_conf_set()	444
7.15.5.6 vtss_restart_conf_end()	445
7.15.5.7 vtss_restart_status_get()	445
7.15.5.8 vtss_restart_conf_get()	446
7.15.5.9 vtss_restart_conf_set()	446
7.16 vtss_api/include/vtss_l2_api.h File Reference	446
7.16.1 Detailed Description	454
7.16.2 Macro Definition Documentation	454
7.16.2.1 VTSS_MAC_ADDRS	454
7.16.2.2 VTSS_MSTIS	454
7.16.2.3 VTSS_MSTI_START	454
7.16.2.4 VTSS_MSTI_END	454
7.16.2.5 VTSS_MSTI_ARRAY_SIZE	455
7.16.2.6 VTSS_VCL_IDS	455
7.16.2.7 VTSS_VCL_ID_START	455
7.16.2.8 VTSS_VCL_ID_END	455
7.16.2.9 VTSS_VCL_ARRAY_SIZE	455
7.16.2.10 VTSS_VCE_ID_LAST	456
7.16.2.11 VTSS_VLAN_TRANS_GROUP_MAX_CNT	456
7.16.2.12 VTSS_VLAN_TRANS_MAX_CNT	456
7.16.2.13 VTSS_VLAN_TRANS_NULL_GROUP_ID	456
7.16.2.14 VTSS_VLAN_TRANS_FIRST_GROUP_ID	456
7.16.2.15 VTSS_VLAN_TRANS_VID_START	457

7.16.2.16 VTSS_VLAN_TRANS_MAX_VLAN_ID	457
7.16.2.17 VTSS_VLAN_TRANS_LAST_GROUP_ID	457
7.16.2.18 VTSS_VLAN_TRANS_VALID_GROUP_CHECK	457
7.16.2.19 VTSS_VLAN_TRANS_VALID_VLAN_CHECK	458
7.16.2.20 VTSS_VLAN_TRANS_NULL_CHECK	458
7.16.2.21 VTSS_VLAN_TRANS_PORT_BF_SIZE	458
7.16.2.22 VTSS_PVLANS	458
7.16.2.23 VTSS_PVLAN_NO_START	459
7.16.2.24 VTSS_PVLAN_NO_END	459
7.16.2.25 VTSS_PVLAN_ARRAY_SIZE	459
7.16.2.26 VTSS_PVLAN_NO_DEFAULT	459
7.16.2.27 VTSS_ERPIS	459
7.16.2.28 VTSS_ERPI_START	460
7.16.2.29 VTSS_ERPI_END	460
7.16.2.30 VTSS_ERPI_ARRAY_SIZE	460
7.16.3 Typedef Documentation	460
7.16.3.1 vtss_vt_id_t	460
7.16.4 Enumeration Type Documentation	460
7.16.4.1 vtss_stp_state_t	460
7.16.4.2 vtss_vlan_port_type_t	461
7.16.4.3 vtss_vlan_tx_tag_t	461
7.16.4.4 vtss_vce_type_t	461
7.16.4.5 vtss_eps_port_type_t	462
7.16.4.6 vtss_eps_selector_t	462
7.16.4.7 vtss_erps_state_t	462
7.16.5 Function Documentation	463
7.16.5.1 vtss_mac_table_add()	463
7.16.5.2 vtss_mac_table_del()	463
7.16.5.3 vtss_mac_table_get()	464
7.16.5.4 vtss_mac_table_get_next()	464

7.16.5.5 vtss_mac_table_age_time_get()	464
7.16.5.6 vtss_mac_table_age_time_set()	465
7.16.5.7 vtss_mac_table_age()	465
7.16.5.8 vtss_mac_table_vlan_age()	466
7.16.5.9 vtss_mac_table_flush()	466
7.16.5.10 vtss_mac_table_port_flush()	466
7.16.5.11 vtss_mac_table_vlan_flush()	467
7.16.5.12 vtss_mac_table_vlan_port_flush()	467
7.16.5.13 vtss_mac_table_status_get()	467
7.16.5.14 vtss_learn_port_mode_get()	468
7.16.5.15 vtss_learn_port_mode_set()	468
7.16.5.16 vtss_port_state_get()	469
7.16.5.17 vtss_port_state_set()	469
7.16.5.18 vtss_stp_port_state_get()	470
7.16.5.19 vtss_stp_port_state_set()	470
7.16.5.20 vtss_mstp_vlan_msti_get()	470
7.16.5.21 vtss_mstp_vlan_msti_set()	471
7.16.5.22 vtss_mstp_port_msti_state_get()	471
7.16.5.23 vtss_mstp_port_msti_state_set()	472
7.16.5.24 vtss_vlan_conf_get()	472
7.16.5.25 vtss_vlan_conf_set()	473
7.16.5.26 vtss_vlan_port_conf_get()	473
7.16.5.27 vtss_vlan_port_conf_set()	473
7.16.5.28 vtss_vlan_port_members_get()	474
7.16.5.29 vtss_vlan_port_members_set()	474
7.16.5.30 vtss_vlan_vid_conf_get()	475
7.16.5.31 vtss_vlan_vid_conf_set()	475
7.16.5.32 vtss_vlan_tx_tag_get()	476
7.16.5.33 vtss_vlan_tx_tag_set()	476
7.16.5.34 vtss_vcl_port_conf_get()	476

7.16.5.35 vtss_vcl_port_conf_set()	477
7.16.5.36 vtss_vce_init()	477
7.16.5.37 vtss_vce_add()	478
7.16.5.38 vtss_vce_del()	478
7.16.5.39 vtss_vlan_trans_group_add()	479
7.16.5.40 vtss_vlan_trans_group_del()	479
7.16.5.41 vtss_vlan_trans_group_get()	479
7.16.5.42 vtss_vlan_trans_group_to_port_set()	480
7.16.5.43 vtss_vlan_trans_group_to_port_get()	480
7.16.5.44 vtss_isolated_vlan_get()	481
7.16.5.45 vtss_isolated_vlan_set()	481
7.16.5.46 vtss_isolated_port_members_get()	482
7.16.5.47 vtss_isolated_port_members_set()	482
7.16.5.48 vtss_pvlan_port_members_get()	482
7.16.5.49 vtss_pvlan_port_members_set()	483
7.16.5.50 vtss_apvlan_port_members_get()	483
7.16.5.51 vtss_apvlan_port_members_set()	484
7.16.5.52 vtss_dgroup_port_conf_get()	484
7.16.5.53 vtss_dgroup_port_conf_set()	485
7.16.5.54 vtss_sflow_port_conf_get()	485
7.16.5.55 vtss_sflow_port_conf_set()	485
7.16.5.56 vtss_sflow_sampling_rate_convert()	486
7.16.5.57 vtss_aggr_port_members_get()	486
7.16.5.58 vtss_aggr_port_members_set()	487
7.16.5.59 vtss_aggr_mode_get()	487
7.16.5.60 vtss_aggr_mode_set()	488
7.16.5.61 vtss_mirror_conf_get()	488
7.16.5.62 vtss_mirror_conf_set()	488
7.16.5.63 vtss_mirror_monitor_port_get()	489
7.16.5.64 vtss_mirror_monitor_port_set()	489

7.16.5.65 vtss_mirror_ingress_ports_get()	490
7.16.5.66 vtss_mirror_ingress_ports_set()	490
7.16.5.67 vtss_mirror_egress_ports_get()	490
7.16.5.68 vtss_mirror_egress_ports_set()	491
7.16.5.69 vtss_mirror_cpu_ingress_get()	491
7.16.5.70 vtss_mirror_cpu_ingress_set()	491
7.16.5.71 vtss_mirror_cpu_egress_get()	492
7.16.5.72 vtss_mirror_cpu_egress_set()	492
7.16.5.73 vtss_uc_flood_members_get()	493
7.16.5.74 vtss_uc_flood_members_set()	493
7.16.5.75 vtss_mc_flood_members_get()	493
7.16.5.76 vtss_mc_flood_members_set()	494
7.16.5.77 vtss_ipv4_mc_flood_members_get()	494
7.16.5.78 vtss_ipv4_mc_flood_members_set()	495
7.16.5.79 vtss_ipv4_mc_add()	495
7.16.5.80 vtss_ipv4_mc_del()	495
7.16.5.81 vtss_ipv6_mc_flood_members_get()	497
7.16.5.82 vtss_ipv6_mc_flood_members_set()	497
7.16.5.83 vtss_ipv6_mc_ctrl_flood_get()	498
7.16.5.84 vtss_ipv6_mc_ctrl_flood_set()	498
7.16.5.85 vtss_ipv6_mc_add()	498
7.16.5.86 vtss_ipv6_mc_del()	499
7.16.5.87 vtss_eps_port_conf_get()	499
7.16.5.88 vtss_eps_port_conf_set()	500
7.16.5.89 vtss_eps_port_selector_get()	500
7.16.5.90 vtss_eps_port_selector_set()	501
7.16.5.91 vtss_erps_vlan_member_get()	501
7.16.5.92 vtss_erps_vlan_member_set()	501
7.16.5.93 vtss_erps_port_state_get()	503
7.16.5.94 vtss_erps_port_state_set()	503

7.17 vtss_api/include/vtss_l3_api.h File Reference	504
7.17.1 Detailed Description	504
7.18 vtss_api/include/vtss_mac10g_api.h File Reference	504
7.18.1 Detailed Description	504
7.19 vtss_api/include/vtss_misc_api.h File Reference	504
7.19.1 Detailed Description	510
7.19.2 Macro Definition Documentation	510
7.19.2.1 VTSS_OS_TIMESTAMP_TYPE	510
7.19.2.2 VTSS_OS_TIMESTAMP	510
7.19.3 Typedef Documentation	511
7.19.3.1 tod_get_ns_cnt_cb_t	511
7.19.4 Enumeration Type Documentation	511
7.19.4.1 vtss_trace_layer_t	511
7.19.4.2 vtss_trace_group_t	511
7.19.4.3 vtss_trace_level_t	512
7.19.4.4 vtss_debug_layer_t	512
7.19.4.5 vtss_debug_group_t	514
7.19.4.6 vtss_gpio_mode_t	515
7.19.4.7 vtss_sgpio_mode_t	515
7.19.4.8 vtss_sgpio_bmode_t	516
7.19.4.9 vtss_irq_t	516
7.19.4.10 vtss_eee_state_select_t	516
7.19.5 Function Documentation	517
7.19.5.1 vtss_trace_conf_get()	517
7.19.5.2 vtss_trace_conf_set()	517
7.19.5.3 vtss_callout_trace_printf()	518
7.19.5.4 vtss_callout_trace_hex_dump()	518
7.19.5.5 vtss_debug_info_get()	519
7.19.5.6 vtss_debug_info_print()	519
7.19.5.7 vtss_callout_lock()	520

7.19.5.8 vtss_callout_unlock()	520
7.19.5.9 vtss_debug_lock()	520
7.19.5.10 vtss_debug_unlock()	521
7.19.5.11 vtss_reg_read()	521
7.19.5.12 vtss_reg_write()	521
7.19.5.13 vtss_reg_write_masked()	522
7.19.5.14 vtss_intr_sticky_clear()	522
7.19.5.15 vtss_chip_id_get()	523
7.19.5.16 vtss_poll_1sec()	523
7.19.5.17 vtss_ptp_event_poll()	524
7.19.5.18 vtss_ptp_event_enable()	524
7.19.5.19 vtss_dev_all_event_poll()	524
7.19.5.20 vtss_dev_all_event_enable()	525
7.19.5.21 vtss_gpio_mode_set()	525
7.19.5.22 vtss_gpio_direction_set()	526
7.19.5.23 vtss_gpio_read()	526
7.19.5.24 vtss_gpio_write()	527
7.19.5.25 vtss_gpio_event_poll()	527
7.19.5.26 vtss_gpio_event_enable()	528
7.19.5.27 vtss_sgpio_conf_get()	528
7.19.5.28 vtss_sgpio_conf_set()	529
7.19.5.29 vtss_sgpio_read()	529
7.19.5.30 vtss_sgpio_event_poll()	529
7.19.5.31 vtss_sgpio_event_enable()	530
7.19.5.32 vtss_intr_cfg()	531
7.19.5.33 vtss_intr_mask_set()	531
7.19.5.34 vtss_intr_status_get()	531
7.19.5.35 vtss_intr_pol_negation()	532
7.19.5.36 vtss_irq_conf_get()	532
7.19.5.37 vtss_irq_conf_set()	533

7.19.5.38 <code>vtss_irq_status_get_and_mask()</code>	533
7.19.5.39 <code>vtss_irq_enable()</code>	533
7.19.5.40 <code>vtss_tod_get_ns_cnt()</code>	534
7.19.5.41 <code>vtss_tod_set_ns_cnt_cb()</code>	534
7.19.5.42 <code>vtss_temp_sensor_init()</code>	534
7.19.5.43 <code>vtss_temp_sensor_get()</code>	535
7.19.5.44 <code>vtss_fan_rotation_get()</code>	535
7.19.5.45 <code>vtss_fan_cool_lvl_set()</code>	536
7.19.5.46 <code>vtss_fan_controller_init()</code>	536
7.19.5.47 <code>vtss_fan_cool_lvl_get()</code>	536
7.19.5.48 <code>vtss_eee_port_conf_set()</code>	537
7.19.5.49 <code>vtss_eee_port_state_set()</code>	537
7.19.5.50 <code>vtss_eee_port_counter_get()</code>	538
7.19.5.51 <code>vtss_debug_reg_check_set()</code>	538
7.20 <code>vtss_api/include/vtss_mpls_api.h</code> File Reference	539
7.20.1 Detailed Description	539
7.21 <code>vtss_api/include/vtss_oam_api.h</code> File Reference	539
7.21.1 Detailed Description	539
7.22 <code>vtss_api/include/vtss_oha_api.h</code> File Reference	539
7.22.1 Detailed Description	540
7.23 <code>vtss_api/include/vtss_os.h</code> File Reference	540
7.23.1 Detailed Description	540
7.24 <code>vtss_api/include/vtss_os_custom.h</code> File Reference	540
7.24.1 Detailed Description	541
7.24.2 Macro Definition Documentation	541
7.24.2.1 <code>uint</code>	541
7.24.2.2 <code>ulong</code>	541
7.24.2.3 <code>VTSS_MSLEEP</code>	541
7.24.2.4 <code>VTSS_MTIMER_START</code>	541
7.24.2.5 <code>VTSS_MTIMER_TIMEOUT</code>	542

7.24.2.6 VTSS_MTIMER_CANCEL	542
7.24.2.7 VTSS_DIV64	542
7.24.2.8 VTSS_MOD64	542
7.24.2.9 VTSS_LABS	543
7.24.2.10 VTSS_LLabs	543
7.24.2.11 VTSS_OS_CTZ	543
7.24.2.12 VTSS_OS_CTZ64	543
7.24.2.13 VTSS_OS_MALLOC	544
7.24.2.14 VTSS_OS_FREE	544
7.24.2.15 VTSS_OS_RAND	544
7.24.3 Typedef Documentation	544
7.24.3.1 vtss_mtimer_t	544
7.25 vtss_api/include/vtss_os_ecos.h File Reference	545
7.25.1 Detailed Description	546
7.25.2 Macro Definition Documentation	546
7.25.2.1 VTSS_MSLEEP	546
7.25.2.2 VTSS_NSLEEP	546
7.25.2.3 VTSS_MTIMER_START	547
7.25.2.4 VTSS_MTIMER_TIMEOUT	547
7.25.2.5 VTSS_MTIMER_CANCEL	547
7.25.2.6 VTSS_TIME_OF_DAY	547
7.25.2.7 VTSS_DIV64	547
7.25.2.8 VTSS_MOD64	548
7.25.2.9 VTSS_LABS	548
7.25.2.10 VTSS_LLabs	548
7.25.2.11 VTSS_OS_CTZ	548
7.25.2.12 VTSS_OS_CTZ64	549
7.25.2.13 VTSS_OS_MALLOC	549
7.25.2.14 VTSS_OS_FREE	549
7.25.2.15 VTSS_OS_RAND	550

7.25.2.16 VTSS_OS_REORDER_BARRIER	550
7.25.2.17 VTSS_OS_COMPILER_ATTRIBUTE_ALIGNED	550
7.25.2.18 VTSS_OS_DCACHE_LINE_SIZE_BYTES	550
7.25.2.19 VTSS_OS_DCACHE_INVALIDATE	551
7.25.2.20 VTSS_OS_DCACHE_FLUSH	551
7.25.2.21 VTSS_OS_VIRT_TO_PHYS	551
7.25.2.22 VTSS_OS_BIG_ENDIAN	551
7.25.2.23 VTSS_OS_NTOHL	552
7.25.2.24 VTSS_OS_SCHEDULER_FLAGS	552
7.25.2.25 VTSS_OS_SCHEDULER_LOCK	552
7.25.2.26 VTSS_OS_SCHEDULER_UNLOCK	552
7.25.2.27 VTSS_OS_INTERRUPT_FLAGS	553
7.25.2.28 VTSS_OS_INTERRUPT_DISABLE	553
7.25.2.29 VTSS_OS_INTERRUPT_RESTORE	553
7.25.3 Typedef Documentation	553
7.25.3.1 vtss_mtimer_t	553
7.25.4 Function Documentation	553
7.25.4.1 llabs()	553
7.25.4.2 vtss_callout_malloc()	554
7.25.4.3 vtss_callout_free()	554
7.26 vtss_api/include/vtss_os_linux.h File Reference	555
7.26.1 Detailed Description	555
7.26.2 Macro Definition Documentation	556
7.26.2.1 VTSS_OS_BIG_ENDIAN	556
7.26.2.2 VTSS_OS_NTOHL	556
7.26.2.3 VTSS_NSLEEP	556
7.26.2.4 VTSS_MSLEEP	557
7.26.2.5 VTSS_MTIMER_START	557
7.26.2.6 VTSS_MTIMER_TIMEOUT	557
7.26.2.7 VTSS_MTIMER_CANCEL	558

7.26.2.8 VTSS_TIME_OF_DAY	558
7.26.2.9 VTSS_OS_SCHEDULER_FLAGS	558
7.26.2.10 VTSS_OS_SCHEDULER_LOCK	558
7.26.2.11 VTSS_OS_SCHEDULER_UNLOCK	559
7.26.2.12 VTSS_DIV64	559
7.26.2.13 VTSS_MOD64	559
7.26.2.14 VTSS_LABS	559
7.26.2.15 VTSS_LLabs	560
7.26.2.16 VTSS_OS_CTZ	560
7.26.2.17 VTSS_OS_CTZ64	561
7.26.2.18 VTSS_OS_MALLOC	561
7.26.2.19 VTSS_OS_FREE	562
7.26.2.20 VTSS_OS_RAND	562
7.27 vtss_api/include/vtss_otp_api.h File Reference	562
7.27.1 Detailed Description	562
7.28 vtss_api/include/vtss_packet_api.h File Reference	562
7.28.1 Detailed Description	565
7.28.2 Macro Definition Documentation	565
7.28.2.1 VTSS_PRIO_SUPER	566
7.28.2.2 VTSS_JR1_PACKET_HDR_SIZE_BYTES	566
7.28.2.3 VTSS_JR2_PACKET_HDR_SIZE_BYTES	566
7.28.2.4 VTSS_SVL_PACKET_HDR_SIZE_BYTES	566
7.28.2.5 VTSS_L26_PACKET_HDR_SIZE_BYTES	566
7.28.2.6 VTSS_PACKET_HDR_SIZE_BYTES	567
7.28.2.7 VTSS_SVL_RX_IFH_SIZE	567
7.28.2.8 VTSS_JR1_RX_IFH_SIZE	567
7.28.2.9 VTSS_L26_RX_IFH_SIZE	567
7.28.2.10 VTSS_JR2_RX_IFH_SIZE	567
7.28.2.11 VTSS_PACKET_TX_IFH_MAX	568
7.28.3 Enumeration Type Documentation	568

7.28.3.1 <code>vtss_packet_filter_t</code>	568
7.28.3.2 <code>vtss_packet_oam_type_t</code>	568
7.28.3.3 <code>vtss_packet_ptp_action_t</code>	569
7.28.3.4 <code>vtss_tag_type_t</code>	569
7.28.3.5 <code>vtss_packet_rx_hints_t</code>	569
7.28.3.6 <code>vtss_packet_tx_vstax_t</code>	570
7.28.4 Function Documentation	571
7.28.4.1 <code>vtss_npi_conf_get()</code>	571
7.28.4.2 <code>vtss_npi_conf_set()</code>	571
7.28.4.3 <code>vtss_packet_rx_conf_get()</code>	572
7.28.4.4 <code>vtss_packet_rx_conf_set()</code>	572
7.28.4.5 <code>vtss_packet_rx_port_conf_get()</code>	572
7.28.4.6 <code>vtss_packet_rx_port_conf_set()</code>	574
7.28.4.7 <code>vtss_packet_rx_frame_get()</code>	574
7.28.4.8 <code>vtss_packet_rx_frame_get_raw()</code>	575
7.28.4.9 <code>vtss_packet_rx_frame_discard()</code>	575
7.28.4.10 <code>vtss_packet_tx_frame_port()</code>	576
7.28.4.11 <code>vtss_packet_tx_frame_port_vlan()</code>	576
7.28.4.12 <code>vtss_packet_tx_frame_vlan()</code>	577
7.28.4.13 <code>vtss_packet_frame_info_init()</code>	577
7.28.4.14 <code>vtss_packet_frame_filter()</code>	577
7.28.4.15 <code>vtss_packet_port_info_init()</code>	578
7.28.4.16 <code>vtss_packet_port_filter_get()</code>	578
7.28.4.17 <code>vtss_packet_rx_hdr_decode()</code>	579
7.28.4.18 <code>vtss_packet_tx_hdr_encode()</code>	579
7.28.4.19 <code>vtss_packet_tx_hdr_compile()</code>	581
7.28.4.20 <code>vtss_packet_tx_frame()</code>	581
7.28.4.21 <code>vtss_packet_tx_info_init()</code>	582
7.28.4.22 <code>vtss_packet_dma_conf_get()</code>	582
7.28.4.23 <code>vtss_packet_dma_conf_set()</code>	583

7.28.4.24 <code>vtss_packet_dma_offset()</code>	583
7.29 <code>vtss_api/include/vtss_pcs_10gbase_r_api.h</code> File Reference	584
7.29.1 Detailed Description	584
7.30 <code>vtss_api/include/vtss_phy_10g_api.h</code> File Reference	584
7.30.1 Detailed Description	584
7.31 <code>vtss_api/include/vtss_phy_api.h</code> File Reference	584
7.31.1 Detailed Description	594
7.31.2 Macro Definition Documentation	594
7.31.2.1 <code>MAX_CFG_BUF_SIZE</code>	594
7.31.2.2 <code>MAX_STAT_BUF_SIZE</code>	594
7.31.2.3 <code>VTSS_PHY_POWER_ACTIPHY_BIT</code>	595
7.31.2.4 <code>VTSS_PHY_POWER_DYNAMIC_BIT</code>	595
7.31.2.5 <code>VTSS_PHY_ACTIPHY_PWR</code>	595
7.31.2.6 <code>VTSS_PHY_LINK_DOWN_PWR</code>	595
7.31.2.7 <code>VTSS_PHY_LINK_UP_FULL_PWR</code>	595
7.31.2.8 <code>VTSS_PHY_RECov_CLK1</code>	596
7.31.2.9 <code>VTSS_PHY_RECov_CLK2</code>	596
7.31.2.10 <code>VTSS_PHY_RECov_CLK_NUM</code>	596
7.31.2.11 <code>VTSS_PHY_PAGE_STANDARD</code>	596
7.31.2.12 <code>VTSS_PHY_PAGE_EXTENDED</code>	596
7.31.2.13 <code>VTSS_PHY_PAGE_EXTENDED_2</code>	597
7.31.2.14 <code>VTSS_PHY_PAGE_EXTENDED_3</code>	597
7.31.2.15 <code>VTSS_PHY_PAGE_EXTENDED_4</code>	597
7.31.2.16 <code>VTSS_PHY_PAGE_GPIO</code>	597
7.31.2.17 <code>VTSS_PHY_PAGE_1588</code>	597
7.31.2.18 <code>VTSS_PHY_PAGE_MACSEC</code>	598
7.31.2.19 <code>VTSS_PHY_PAGE_TEST</code>	598
7.31.2.20 <code>VTSS_PHY_PAGE_TR</code>	598
7.31.2.21 <code>VTSS_PHY_PAGE_0x2DAF</code>	598
7.31.2.22 <code>VTSS_PHY_REG_STANDARD</code>	598

7.31.2.23 VTSS_PHY_REG_EXTENDED	599
7.31.2.24 VTSS_PHY_REG_GPIO	599
7.31.2.25 VTSS_PHY_REG_TEST	599
7.31.2.26 VTSS_PHY_REG_TR	599
7.31.2.27 VTSS_PHY_LINK_LOS_EV	599
7.31.2.28 VTSS_PHY_LINK_FFAIL_EV	600
7.31.2.29 VTSS_PHY_LINK_AMS_EV	600
7.31.2.30 VTSS_PHY_LINK_SPEED_STATE_CHANGE_EV	600
7.31.2.31 VTSS_PHY_LINK_FDX_STATE_CHANGE_EV	600
7.31.2.32 VTSS_PHY_LINK_AUTO_NEG_ERROR_EV	600
7.31.2.33 VTSS_PHY_LINK_AUTO_NEG_COMPLETE_EV	601
7.31.2.34 VTSS_PHY_LINK_INLINE_POW_DEV_DETECT_EV	601
7.31.2.35 VTSS_PHY_LINK_SYMBOL_ERR_INT_EV	601
7.31.2.36 VTSS_PHY_LINK_TX_FIFO_OVERFLOW_INT_EV	601
7.31.2.37 VTSS_PHY_LINK_RX_FIFO_OVERFLOW_INT_EV	601
7.31.2.38 VTSS_PHY_LINK_FALSE_CARRIER_INT_EV	602
7.31.2.39 VTSS_PHY_LINK_LINK_SPEED_DS_DETECT_EV	602
7.31.2.40 VTSS_PHY_LINK_MASTER_SLAVE_RES_ERR_EV	602
7.31.2.41 VTSS_PHY_LINK_RX_ER_INT_EV	602
7.31.2.42 VTSS_PHY_LINK_EXTENDED_REG_INT_EV	602
7.31.2.43 VTSS_PHY_LINK_WAKE_ON_LAN_INT_EV	603
7.31.2.44 VTSS_PHY_LINK_EXT_EEE_WAKE_ERR_EV	603
7.31.2.45 VTSS_PHY_LINK_EXT_EEE_WAIT_TS_EV	603
7.31.2.46 VTSS_PHY_LINK_EXT_EEE_WAIT_RX_TQ_EV	603
7.31.2.47 VTSS_PHY_LINK_EXT_EEE_LINKFAIL_EV	603
7.31.2.48 VTSS_PHY_LINK_EXT_RR_SW_COMPL_EV	604
7.31.2.49 VTSS_PHY_LINK_EXT_MACSEC_HOST_MAC_EV	604
7.31.2.50 VTSS_PHY_LINK_EXT_MACSEC_LINE_MAC_EV	604
7.31.2.51 VTSS_PHY_LINK_EXT_MACSEC_FC_BUFF_EV	604
7.31.2.52 VTSS_PHY_LINK_EXT_MACSEC_INGRESS_EV	604

7.31.2.53 VTSS_PHY_LINK_EXT_MACSEC_EGRESS_EV	605
7.31.2.54 VTSS_PHY_LINK_EXT_MEM_INT_RING_EV	605
7.31.2.55 MAX_WOL_MAC_ADDR_SIZE	605
7.31.2.56 MAX_WOL_PASSWD_SIZE	605
7.31.2.57 MIN_WOL_PASSWD_SIZE	605
7.31.3 Typedef Documentation	606
7.31.3.1 vtss_phy_recov_clk_t	606
7.31.3.2 vtss_phy_led_intensity	606
7.31.4 Enumeration Type Documentation	606
7.31.4.1 vtss_phy_led_mode_t	606
7.31.4.2 vtss_phy_media_interface_t	607
7.31.4.3 vtss_phy_mdio_t	608
7.31.4.4 rgmii_skew_delay_psec_t	608
7.31.4.5 vtss_phy_forced_reset_t	609
7.31.4.6 vtss_phy_pkt_mode_t	609
7.31.4.7 vtss_phy_mode_t	609
7.31.4.8 vtss_phy_fast_link_fail_t	610
7.31.4.9 vtss_phy_sigdet_polarity_t	610
7.31.4.10 vtss_phy_unidirectional_t	610
7.31.4.11 vtss_phy_mac_serdes_pcs_sgmii_pre	610
7.31.4.12 vtss_phy_media_rem_fault_t	612
7.31.4.13 vtss_phy_media_force_ams_sel_t	612
7.31.4.14 vtss_phy_clk_source_t	613
7.31.4.15 vtss_phy_freq_t	613
7.31.4.16 vtss_phy_clk_squelch	613
7.31.4.17 vtss_phy_gpio_mode_t	614
7.31.4.18 lb_type	614
7.31.4.19 vtss_wol_passwd_len_type_t	615
7.31.4.20 vtss_fefi_mode_t	615
7.31.5 Function Documentation	615

7.31.5.1 vtss_phy_pre_reset()	615
7.31.5.2 vtss_phy_post_reset()	616
7.31.5.3 vtss_phy_pre_system_reset()	616
7.31.5.4 vtss_phy_reset()	617
7.31.5.5 vtss_phy_reset_get()	617
7.31.5.6 vtss_phy_chip_temp_get()	617
7.31.5.7 vtss_phy_chip_temp_init()	618
7.31.5.8 vtss_phy_conf_get()	618
7.31.5.9 vtss_phy_conf_set()	619
7.31.5.10 vtss_phy_status_get()	619
7.31.5.11 vtss_phy_cl37_lp_abil_get()	620
7.31.5.12 vtss_phy_id_get()	620
7.31.5.13 vtss_phy_conf_1g_get()	620
7.31.5.14 vtss_phy_conf_1g_set()	621
7.31.5.15 vtss_phy_status_1g_get()	621
7.31.5.16 vtss_phy_power_conf_get()	622
7.31.5.17 vtss_phy_power_conf_set()	622
7.31.5.18 vtss_phy_power_status_get()	623
7.31.5.19 vtss_phy_clock_conf_set()	623
7.31.5.20 vtss_phy_clock_conf_get()	623
7.31.5.21 vtss_phy_i2c_read()	624
7.31.5.22 vtss_phy_i2c_write()	625
7.31.5.23 vtss_phy_read()	625
7.31.5.24 vtss_phy_read_page()	626
7.31.5.25 vtss_phy_mmd_read()	626
7.31.5.26 vtss_phy_mmd_write()	627
7.31.5.27 vtss_phy_write()	627
7.31.5.28 vtss_phy_write_masked()	628
7.31.5.29 vtss_phy_write_masked_page()	628
7.31.5.30 vtss_phy_gpio_mode()	629

7.31.5.31 vtss_phy_gpio_get()	629
7.31.5.32 vtss_phy_gpio_set()	630
7.31.5.33 vtss_phy_veriphy_start()	630
7.31.5.34 vtss_phy_veriphy_get()	631
7.31.5.35 vtss_phy_led_mode_set()	631
7.31.5.36 vtss_phy_led_intensity_set()	632
7.31.5.37 vtss_phy_led_intensity_get()	632
7.31.5.38 vtss_phy_enhanced_led_control_init()	632
7.31.5.39 vtss_phy_enhanced_led_control_init_get()	633
7.31.5.40 vtss_phy_coma_mode_disable()	633
7.31.5.41 vtss_phy_coma_mode_enable()	634
7.31.5.42 vga_adc_debug()	634
7.31.5.43 vtss_phy_port_eee_capable()	634
7.31.5.44 vtss_phy_eee_ena()	635
7.31.5.45 vtss_phy_eee_conf_get()	635
7.31.5.46 vtss_phy_eee_conf_set()	636
7.31.5.47 vtss_phy_eee_power_save_state_get()	636
7.31.5.48 vtss_phy_eee_link_partner_advertisements_get()	637
7.31.5.49 vtss_phy_event_enable_set()	637
7.31.5.50 vtss_phy_event_enable_get()	638
7.31.5.51 vtss_phy_event_poll()	638
7.31.5.52 vtss_squelch_workaround()	638
7.31.5.53 vtss_phy_csr_wr()	639
7.31.5.54 vtss_phy_csr_rd()	639
7.31.5.55 vtss_phy_statistic_get()	641
7.31.5.56 vtss_phy_do_page_chk_set()	641
7.31.5.57 vtss_phy_do_page_chk_get()	642
7.31.5.58 vtss_phy_loopback_set()	642
7.31.5.59 vtss_phy_loopback_get()	642
7.31.5.60 vtss_phy_is_8051_crc_ok()	643

7.31.5.61 <code>vtss_phy_cfg_ob_post0()</code>	643
7.31.5.62 <code>vtss_phy_cfg_ib_cterm()</code>	644
7.31.5.63 <code>vtss_phy_atom12_patch_settings_get()</code>	644
7.31.5.64 <code>vtss_phy_reg_decode_status()</code>	645
7.31.5.65 <code>vtss_phy_flowcontrol_decode_status()</code>	645
7.31.5.66 <code>vtss_phy_debug_stat_print()</code>	646
7.31.5.67 <code>vtss_phy_warm_start_failed_get()</code>	646
7.31.5.68 <code>vtss_phy_debug_phyinfo_print()</code>	647
7.31.5.69 <code>vtss_phy_debug_register_dump()</code>	647
7.31.5.70 <code>vtss_phy_detect_base_ports()</code>	648
7.31.5.71 <code>vtss_phy_ext_connector_loopback()</code>	648
7.31.5.72 <code>vtss_phy_serdes_sgmii_loopback()</code>	648
7.31.5.73 <code>vtss_phy_serdes_fmedia_loopback()</code>	649
7.31.5.74 <code>vtss_phy_debug_regdump_print()</code>	649
7.31.5.75 <code>vtss_phy_wol_enable()</code>	650
7.31.5.76 <code>vtss_phy_wol_conf_get()</code>	650
7.31.5.77 <code>vtss_phy_wol_conf_set()</code>	651
7.31.5.78 <code>vtss_phy_patch_settings_get()</code>	651
7.31.5.79 <code>vtss_phy_serdes6g_rcpll_status_get()</code>	652
7.31.5.80 <code>vtss_phy_serdes1g_rcpll_status_get()</code>	652
7.31.5.81 <code>vtss_phy_lcpll_status_get()</code>	653
7.31.5.82 <code>vtss_phy_reset_lcpll()</code>	653
7.31.5.83 <code>vtss_phy_sd6g_ob_post_rd()</code>	654
7.31.5.84 <code>vtss_phy_sd6g_ob_post_wr()</code>	654
7.31.5.85 <code>vtss_phy_sd6g_ob_lev_rd()</code>	655
7.31.5.86 <code>vtss_phy_sd6g_ob_lev_wr()</code>	655
7.31.5.87 <code>vtss_phy_mac_media_inhibit_odd_start()</code>	656
7.31.5.88 <code>vtss_phy_fefi_get()</code>	656
7.31.5.89 <code>vtss_phy_fefi_set()</code>	656
7.31.5.90 <code>vtss_phy_fefi_detect()</code>	657

7.31.5.91 <code>vtss_phy_mse_100m_get()</code>	657
7.31.5.92 <code>vtss_phy_mse_1000m_get()</code>	658
7.31.5.93 <code>vtss_phy_read_tr_addr()</code>	658
7.31.5.94 <code>vtss_phy_is_viper_revB()</code>	659
7.31.5.95 <code>vtss_phy_ext_event_poll()</code>	659
7.31.5.96 <code>vtss_phy_status_inst_poll()</code>	660
7.31.5.97 <code>vtss_phy_macsec_csr_sd6g_rd()</code>	660
7.31.5.98 <code>vtss_phy_macsec_csr_sd6g_wr()</code>	661
7.31.5.99 <code>vtss_phy_sd6g_mac_serdes_conf()</code>	661
7.32 <code>vtss_api/include/vtss_phy_ts_api.h</code> File Reference	661
7.32.1 Detailed Description	662
7.33 <code>vtss_api/include/vtss_port_api.h</code> File Reference	662
7.33.1 Detailed Description	664
7.33.2 Macro Definition Documentation	664
7.33.2.1 <code>CHIP_PORT_UNUSED</code>	664
7.33.2.2 <code>VTSS_FRAME_GAP_DEFAULT</code>	664
7.33.2.3 <code>VTSS_MAX_FRAME_LENGTH_STANDARD</code>	665
7.33.2.4 <code>VTSS_MAX_FRAME_LENGTH_MAX [1/2]</code>	665
7.33.2.5 <code>VTSS_MAX_FRAME_LENGTH_MAX [2/2]</code>	665
7.33.3 Enumeration Type Documentation	665
7.33.3.1 <code>vtss_miim_controller_t</code>	665
7.33.3.2 <code>vtss_internal_bw_t</code>	666
7.33.3.3 <code>vtss_port_clause_37_remote_fault_t</code>	666
7.33.3.4 <code>vtss_port_max_tags_t</code>	666
7.33.3.5 <code>vtss_port_loop_t</code>	667
7.33.3.6 <code>vtss_port_forward_t</code>	667
7.33.4 Function Documentation	667
7.33.4.1 <code>vtss_port_map_set()</code>	667
7.33.4.2 <code>vtss_port_map_get()</code>	668
7.33.4.3 <code>vtss_port_clause_37_control_get()</code>	668

7.33.4.4 <code>vtss_port_clause_37_control_set()</code>	669
7.33.4.5 <code>vtss_port_conf_set()</code>	669
7.33.4.6 <code>vtss_port_conf_get()</code>	670
7.33.4.7 <code>vtss_port_status_get()</code>	670
7.33.4.8 <code>vtss_port_counters_update()</code>	670
7.33.4.9 <code>vtss_port_counters_clear()</code>	671
7.33.4.10 <code>vtss_port_counters_get()</code>	671
7.33.4.11 <code>vtss_port_basic_counters_get()</code>	672
7.33.4.12 <code>vtss_port_forward_state_get()</code>	672
7.33.4.13 <code>vtss_port_forward_state_set()</code>	673
7.33.4.14 <code>vtss_miim_read()</code>	673
7.33.4.15 <code>vtss_miim_write()</code>	674
7.34 <code>vtss_api/include/vtss_qos_api.h</code> File Reference	674
7.34.1 Detailed Description	676
7.34.2 Macro Definition Documentation	676
7.34.2.1 <code>VTSS_PORT_POLICERS</code>	676
7.34.2.2 <code>VTSS_PORT_POLICER_CPU_QUEUES</code>	677
7.34.2.3 <code>VTSS_QCL_IDS</code>	677
7.34.2.4 <code>VTSS_QCL_ID_START</code>	677
7.34.2.5 <code>VTSS_QCL_ID_END</code>	677
7.34.2.6 <code>VTSS_QCL_ARRAY_SIZE</code>	677
7.34.2.7 <code>VTSS_QCE_ID_LAST</code>	678
7.34.3 Enumeration Type Documentation	678
7.34.3.1 <code>vtss_tag_remark_mode_t</code>	678
7.34.3.2 <code>vtss_dscp_mode_t</code>	678
7.34.3.3 <code>vtss_dscp_emode_t</code>	678
7.34.3.4 <code>vtss_qce_type_t</code>	679
7.34.4 Function Documentation	679
7.34.4.1 <code>vtss_qos_conf_get()</code>	679
7.34.4.2 <code>vtss_qos_conf_set()</code>	680

7.34.4.3 vtss_mep_policer_conf_get()	680
7.34.4.4 vtss_mep_policer_conf_set()	681
7.34.4.5 vtss_qos_port_conf_get()	681
7.34.4.6 vtss_qos_port_conf_set()	681
7.34.4.7 vtss_qce_init()	682
7.34.4.8 vtss_qce_add()	682
7.34.4.9 vtss_qce_del()	683
7.35 vtss_api/include/vtss_rab_api.h File Reference	683
7.35.1 Detailed Description	683
7.36 vtss_api/include/vtss_security_api.h File Reference	683
7.36.1 Detailed Description	686
7.36.2 Macro Definition Documentation	686
7.36.2.1 VTSS_ACE_ID_LAST	686
7.36.3 Enumeration Type Documentation	686
7.36.3.1 vtss_auth_state_t	686
7.36.3.2 vtss_acl_port_action_t	687
7.36.3.3 vtss_acl_ptp_action_t	687
7.36.3.4 vtss_ace_type_t	687
7.36.3.5 vtss_ace_bit_t	688
7.36.4 Function Documentation	688
7.36.4.1 vtss_auth_port_state_get()	688
7.36.4.2 vtss_auth_port_state_set()	688
7.36.4.3 vtss_acl_policer_conf_get()	689
7.36.4.4 vtss_acl_policer_conf_set()	689
7.36.4.5 vtss_acl_port_conf_get()	690
7.36.4.6 vtss_acl_port_conf_set()	690
7.36.4.7 vtss_acl_port_counter_get()	691
7.36.4.8 vtss_acl_port_counter_clear()	691
7.36.4.9 vtss_ace_init()	691
7.36.4.10 vtss_ace_add()	692

7.36.4.11 vtss_ace_del()	692
7.36.4.12 vtss_ace_counter_get()	693
7.36.4.13 vtss_ace_counter_clear()	693
7.37 vtss_api/include/vtss_sfi4_api.h File Reference	693
7.37.1 Detailed Description	694
7.38 vtss_api/include/vtss_sync_api.h File Reference	694
7.38.1 Detailed Description	695
7.38.2 Macro Definition Documentation	695
7.38.2.1 VTSS_SYNCE_CLK_A	695
7.38.2.2 VTSS_SYNCE_CLK_B	695
7.38.2.3 VTSS_SYNCE_CLK_MAX	695
7.38.3 Enumeration Type Documentation	695
7.38.3.1 vtss_syncce_divider_t	695
7.38.4 Function Documentation	696
7.38.4.1 vtss_syncce_clock_out_set()	696
7.38.4.2 vtss_syncce_clock_out_get()	696
7.38.4.3 vtss_syncce_clock_in_set()	697
7.38.4.4 vtss_syncce_clock_in_get()	697
7.39 vtss_api/include/vtss_tfi5_api.h File Reference	697
7.39.1 Detailed Description	698
7.40 vtss_api/include/vtss_ts_api.h File Reference	698
7.40.1 Detailed Description	701
7.40.2 Function Documentation	701
7.40.2.1 vtss_ts_timeofday_set()	701
7.40.2.2 vtss_ts_timeofday_set_delta()	701
7.40.2.3 vtss_ts_timeofday_offset_set()	702
7.40.2.4 vtss_ts_adjtimer_one_sec()	702
7.40.2.5 vtss_ts_ongoing_adjustment()	703
7.40.2.6 vtss_ts_timeofday_get()	703
7.40.2.7 vtss_ts_timeofday_next_pps_get()	703

7.40.2.8 vtss_ts_adjtimer_set()	704
7.40.2.9 vtss_ts_adjtimer_get()	704
7.40.2.10 vtss_ts_freq_offset_get()	705
7.40.2.11 vtss_ts_external_clock_mode_get()	705
7.40.2.12 vtss_ts_external_clock_mode_set()	705
7.40.2.13 vtss_ts_external_clock_saved_get()	706
7.40.2.14 vtss_ts_ingress_latency_set()	706
7.40.2.15 vtss_ts_ingress_latency_get()	707
7.40.2.16 vtss_ts_p2p_delay_set()	707
7.40.2.17 vtss_ts_p2p_delay_get()	707
7.40.2.18 vtss_ts_egress_latency_set()	708
7.40.2.19 vtss_ts_egress_latency_get()	708
7.40.2.20 vtss_ts_delay_asymmetry_set()	709
7.40.2.21 vtss_ts_delay_asymmetry_get()	709
7.40.2.22 vtss_ts_operation_mode_set()	710
7.40.2.23 vtss_ts_operation_mode_get()	710
7.40.2.24 vtss_ts_internal_mode_set()	710
7.40.2.25 vtss_ts_internal_mode_get()	711
7.40.2.26 vtss_tx_timestamp_update()	711
7.40.2.27 vtss_rx_timestamp_get()	712
7.40.2.28 vtss_rx_timestamp_id_release()	712
7.40.2.29 vtss_rx_master_timestamp_get()	712
7.40.2.30 vtss_tx_timestamp_idx_alloc()	713
7.40.2.31 vtss_timestamp_age()	713
7.40.2.32 vtss_ts_status_change()	714
7.41 vtss_api/include/vtss_upi_api.h File Reference	714
7.41.1 Detailed Description	714
7.42 vtss_api/include/vtss_wis_api.h File Reference	714
7.42.1 Detailed Description	714
7.43 vtss_api/include/vtss_xaui_api.h File Reference	714
7.43.1 Detailed Description	715
7.44 vtss_api/include/vtss_xfi_api.h File Reference	715
7.44.1 Detailed Description	715
Index	717

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

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

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

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

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

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>tag_vtss_fdma_list</code>	Software DCB structure	27
<code>vtss_ace_frame_arp_t</code>	Frame data for VTSS_ACE_TYPE_ARP	31
<code>vtss_ace_frame_etype_t</code>	Frame data for VTSS_ACE_TYPE_ETYPE	33
<code>vtss_ace_frame_ipv4_t</code>	Frame data for VTSS_ACE_TYPE_IPV4	35
<code>vtss_ace_frame_ipv6_t</code>	Frame data for VTSS_ACE_TYPE_IPV6	40
<code>vtss_ace_frame_llc_t</code>	Frame data for VTSS_ACE_TYPE_LLCC	44
<code>vtss_ace_frame_snap_t</code>	Frame data for VTSS_ACE_TYPE_SNAP	45
<code>vtss_ace_ptp_t</code>	PTP header filtering	46
<code>vtss_ace_sip_smac_t</code>	SIP/SMAC filtering	47
<code>vtss_ace_t</code>	Access Control Entry	48
<code>vtss_ace_vlan_t</code>	ACE VLAN information	52
<code>vtss_acl_action_t</code>	ACL Action	54
<code>vtss_acl_policer_conf_t</code>	ACL policer configuration	57
<code>vtss_acl_port_conf_t</code>	ACL port configuration	58
<code>vtss_aggr_mode_t</code>	Aggregation traffic distribution mode	59
<code>vtss_aneg_t</code>	Auto negotiation struct	60

<code>vtss_api_lock_t</code>	API lock structure	61
<code>vtss_basic_counters_t</code>	Basic counters structure	63
<code>vtss_chip_id_t</code>	Chip ID	63
<code>vtss_counter_pair_t</code>	Counter pair	64
<code>vtss_debug_info_t</code>	Debug information structure	65
<code>vtss_debug_lock_t</code>	API debug lock structure	67
<code>vtss_dgroup_port_conf_t</code>	Destination group port configuration	68
<code>vtss_dlb_policer_conf_t</code>	Dual leaky buckets policer configuration	69
<code>vtss_ece_action_t</code>	ECE action	71
<code>vtss_ece_frame_ipv4_t</code>	ECE IPv4 information	73
<code>vtss_ece_frame_ipv6_t</code>	ECE IPv6 information	75
<code>vtss_ece_key_t</code>	ECE key	76
<code>vtss_ece_mac_t</code>	ECE MAC information	78
<code>vtss_ece_outer_tag_t</code>	ECE outer tag	79
<code>vtss_ece_t</code>	EVC Control Entry	81
<code>vtss_ece_tag_t</code>	ECE tag information	82
<code>vtss_eee_port_conf_t</code>	EEE port configuration	83
<code>vtss_eee_port_counter_t</code>	EEE port counters (JR only)	85
<code>vtss_eee_port_state_t</code>	EEE port state (JR only)	87
<code>vtss_eps_port_conf_t</code>	Port protection configuration	87
<code>vtss_evc_conf_t</code>	EVC configuration (excluding UNIs)	88
<code>vtss_evc_inner_tag_t</code>	EVC inner tag	90
<code>vtss_evc_pb_conf_t</code>	PB specific EVC configuration	91
<code>vtss_evc_port_conf_t</code>	EVC port configuration	93
<code>vtss_fan_conf_t</code>	Fan specifications	94
<code>vtss_fdma_cfg_t</code>	FDMA configuration structure	96
<code>vtss_fdma_ch_cfg_t</code>	Channel configuration structure	101
<code>vtss_fdma_throttle_cfg_t</code>		105
<code>vtss_fdma_tx_info_t</code>	FDMA Injection Properties	107

vtss_init_conf_t	Initialization configuration	111
vtss_inst_create_t	Create structure	115
vtss_intr_t	Interrupt source structure	115
vtss_ip_addr_t	Either an IPv4 or IPv6 address	116
vtss_ip_network_t	IPv6 network	117
vtss_ipv4_network_t	IPv4 network	118
vtss_ipv4_uc_t	IPv4 unicast routing entry	119
vtss_ipv6_network_t	IPv6 network	120
vtss_ipv6_t	IPv6 address/mask	121
vtss_ipv6_uc_t	IPv6 routing entry	122
vtss_irq_conf_t	Interrupt configuration options	123
vtss_irq_status_t	Interrupt status structure	124
vtss_l3_counters_t	Routing interface statics counter	125
vtss_lcpll_status_t	Structure for Get PHY LC-PLL status	127
vtss_learn_mode_t	Learning mode	129
vtss_mac_t	MAC Address	130
vtss_mac_table_entry_t	MAC address entry	131
vtss_mac_table_status_t	MAC address table status	133
vtss_mce_action_t	MCE action	134
vtss_mce_key_t	MCE key	136
vtss_mce_t	MEP Control Entry	137
vtss_mirror_conf_t	Mirror configuration	138
vtss_mtimer_t	Timer structure	139
vtss_npi_conf_t	NPI configuration	140
vtss_os_timestamp_t	141
vtss_packet_dma_conf_t	142
vtss_packet_frame_info_t	Information about frame	143
vtss_packet_port_filter_t	Packet information for each port	144
vtss_packet_port_info_t	Port info structure	145
vtss_packet_rx_conf_t	CPU Rx configuration	147

<code>vtss_packet_rx_header_t</code>	System frame header describing received frame	148
<code>vtss_packet_rx_info_t</code>	Decoded extraction header properties	150
<code>vtss_packet_rx_meta_t</code>	Input structure to <code>vtss_packet_rx_hdr_decode()</code>	159
<code>vtss_packet_rx_port_conf_t</code>	Packet registration per port	163
<code>vtss_packet_rx_queue_conf_t</code>	CPU Rx queue configuration	164
<code>vtss_packet_rx_queue_map_t</code>	CPU Rx queue map	165
<code>vtss_packet_rx_queue_npi_conf_t</code>	CPU Rx queue NPI configuration	167
<code>vtss_packet_rx_reg_t</code>	CPU Rx packet registration	168
<code>vtss_packet_tx_ifh_t</code>	Compiled Tx Frame Header	170
<code>vtss_packet_tx_info_t</code>	Injection Properties	171
<code>vtss_phy_aneg_t</code>	PHY auto negotiation advertisement	180
<code>vtss_phy_clock_conf_t</code>	PHY clock configuration	182
<code>vtss_phy_conf_1g_t</code>	PHY 1G configuration	184
<code>vtss_phy_conf_t</code>	PHY configuration	185
<code>vtss_phy_eee_conf_t</code>	EEE configuration	188
<code>vtss_phy_enhanced_led_control_t</code>	Enhanced LED control	188
<code>vtss_phy_forced_t</code>	PHY forced mode configuration	190
<code>vtss_phy_led_mode_select_t</code>	LED model selection	191
<code>vtss_phy_loopback_t</code>	1G Phy loopbacks	192
<code>vtss_phy_mac_serdes_pcs_ctrl_t</code>	PHY MAC SerDes PCS Control, Reg16E3	194
<code>vtss_phy_media_serdes_pcs_ctrl_t</code>	PHY MEDIA SerDes PCS Control, Reg23E3	197
<code>vtss_phy_power_conf_t</code>	PHY power configuration	200
<code>vtss_phy_power_status_t</code>	PHY power status	200
<code>vtss_phy_reset_conf_t</code>	PHY reset structure	201
<code>vtss_phy_rgmiil_conf_t</code>	PHY RGMIIL configuration	203
<code>vtss_phy_statistic_t</code>	Phy statistic information	204
<code>vtss_phy_status_1g_t</code>	PHY 1G status	206
<code>vtss_phy_tbi_conf_t</code>	PHY TBI configuration	207
<code>vtss_phy_type_t</code>	Phy type information	208

<code>vtss_phy_veriphy_result_t</code>	VeriPHY result	210
<code>vtss_phy_wol_conf_t</code>	Structure for Get/Set Wake-On-LAN configuration	211
<code>vtss_pi_conf_t</code>	PI configuration	212
<code>vtss_policer_ext_t</code>	Policer Extensions	214
<code>vtss_policer_t</code>	Policer	214
<code>vtss_port_bridge_counters_t</code>	Port bridge counter structure (RFC 4188)	215
<code>vtss_port_clause_37_adv_t</code>	Advertisement control data for Clause 37 aneg	216
<code>vtss_port_clause_37_control_t</code>	Auto-negotiation control parameter struct	218
<code>vtss_port_conf_t</code>	Port configuration structure	219
<code>vtss_port_counters_t</code>	Port counter structure	223
<code>vtss_port_ethernet_like_counters_t</code>	Ethernet-like Interface counter structure (RFC 3635)	225
<code>vtss_port_evc_counters_t</code>	EVC counters	226
<code>vtss_port_flow_control_conf_t</code>	Flow control setup	228
<code>vtss_port_frame_gaps_t</code>	Inter frame gap structure	229
<code>vtss_port_if_group_counters_t</code>	Interfaces Group counter structure (RFC 2863)	231
<code>vtss_port_map_t</code>	Port map structure	234
<code>vtss_port_proprietary_counters_t</code>	Port proprietary counter structure	236
<code>vtss_port_rmon_counters_t</code>	RMON counter structure (RFC 2819)	236
<code>vtss_port_serdes_conf_t</code>	SFI Serdes configuration	243
<code>vtss_port_sgmii_aneg_t</code>	Advertisement control data for SGMII aneg	244
<code>vtss_port_status_t</code>	Port status parameter struct	246
<code>vtss_qce_action_t</code>	QCE action	249
<code>vtss_qce_frame_etype_t</code>	Frame data for VTSS_QCE_TYPE_ETYPE	252
<code>vtss_qce_frame_ip4_t</code>	Frame data for VTSS_QCE_TYPE_IPV4	252
<code>vtss_qce_frame_ip6_t</code>	Frame data for VTSS_QCE_TYPE_IPV6	254
<code>vtss_qce_frame_llc_t</code>	Frame data for VTSS_QCE_TYPE_LLC	256
<code>vtss_qce_frame_snap_t</code>	Frame data for VTSS_QCE_TYPE_SNAP	257
<code>vtss_qce_key_t</code>	QCE key	257
<code>vtss_qce_mac_t</code>	QCE MAC information	260

<code>vtss_qce_t</code>	QoS Control Entry	261
<code>vtss_qce_tag_t</code>	QCE tag information	262
<code>vtss_qos_conf_t</code>	All parameters below are defined per chip	264
<code>vtss_qos_port_conf_t</code>	QoS setup per port	268
<code>vtss_rcpll_status_t</code>	Structure for Get PHY RC-PLL status	274
<code>vtss_restart_status_t</code>	Restart status	276
<code>vtss_routing_entry_t</code>	Routing entry	277
<code>vtss_secure_on_passwd_t</code>	Structure for Wake-On-LAN Secure-On Password	278
<code>vtss_serdes_macro_conf_t</code>	Serdes macro configuration	279
<code>vtss_sflow_port_conf_t</code>	SFlow configuration structure	280
<code>vtss_sgpi_o_conf_t</code>	SGPIO configuration for a group	281
<code>vtss_sgpi_o_port_conf_t</code>	SGPIO port configuration	283
<code>vtss_sgpi_o_port_data_t</code>	SGPIO read data for a port	284
<code>vtss_shaper_t</code>	Shaper	285
<code>vtss_sync_e_clock_in_t</code>	Struct containing configuration selecting the recovered input clock port, to be delivered to a selected output clock port	286
<code>vtss_sync_e_clock_out_t</code>	Struct containing configuration for a recovered clock output port	287
<code>vtss_tci_t</code>	Tag Control Information (according to IEEE 802.1Q)	288
<code>vtss_timeofday_t</code>	Time of day structure	289
<code>vtss_timestamp_t</code>	Time stamp in seconds and nanoseconds	290
<code>vtss_trace_conf_t</code>	Trace group configuration	291
<code>vtss_ts_ext_clock_mode_t</code>	External clock output configuration	292
<code>vtss_ts_id_t</code>	Timestamp identifier	293
<code>vtss_ts_internal_mode_t</code>	Hardware timestamping format mode for internal ports	294
<code>vtss_ts_operation_mode_t</code>	Timestamp operation	295
<code>vtss_ts_timestamp_alloc_t</code>	Timestamp allocation	295
<code>vtss_ts_timestamp_t</code>	Timestamp structure	296
<code>vtss_vcap_ip_t</code>	VCAP IPv4 address value and mask	298
<code>vtss_vcap_u128_t</code>	VCAP 128 bit value and mask	299

vtss_vcap_u16_t	VCAP 16 bit value and mask	300
vtss_vcap_u24_t	VCAP 24 bit value and mask	301
vtss_vcap_u32_t	VCAP 32 bit value and mask	302
vtss_vcap_u40_t	VCAP 40 bit value and mask	303
vtss_vcap_u48_t	VCAP 48 bit value and mask	304
vtss_vcap_u8_t	VCAP 8 bit value and mask	305
vtss_vcap_udp_tcp_t	VCAP UDP/TCP port range	306
vtss_vcap_vid_t	VCAP VLAN ID value and mask	307
vtss_vcap_vr_t	VCAP universal value or range	308
vtss_vce_action_t	VCE Action	310
vtss_vce_frame_etype_t	Frame data for VTSS_VCE_TYPE_ETYPE	311
vtss_vce_frame_ipv4_t	Frame data for VTSS_VCE_TYPE_IPV4	312
vtss_vce_frame_ipv6_t	Frame data for VTSS_VCE_TYPE_IPV6	314
vtss_vce_frame_llc_t	Frame data for VTSS_VCE_TYPE_LLCC	315
vtss_vce_frame_snap_t	Frame data for VTSS_VCE_TYPE_SNAP	316
vtss_vce_key_t	VCE Key	317
vtss_vce_mac_t	VCE MAC header information	319
vtss_vce_t	VLAN Control Entry	320
vtss_vce_tag_t	VCE tag information	321
vtss_vcl_port_conf_t	VCL port configuration	323
vtss_vid_mac_t	MAC Address in specific VLAN	324
vtss_vlan_conf_t	VLAN configuration	325
vtss_vlan_port_conf_t	VLAN port configuration	326
vtss_vlan_tag_t		327
vtss_vlan_trans_grp2vlan_conf_t	VLAN translation group-to-VLAN configuration	329
vtss_vlan_trans_port2grp_conf_t	VLAN translation port-to-group configuration	330
vtss_vlan_vid_conf_t	VLAN ID configuration	331
vtss_wol_mac_addr_t	Structure for Wake-On-LAN MAC Address	332

Chapter 5

File Index

5.1 File List

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

vtss_api/include/ vtss_ae_api.h	
Ae API	403
vtss_api/include/ vtss_afi_api.h	
AFI API	403
vtss_api/include/ vtss_aneg_api.h	
ANEG API	403
vtss_api/include/ vtss_api.h	
Vitesse API main header file	404
vtss_api/include/ vtss_evc_api.h	
EVC API	404
vtss_api/include/ vtss_fdma_api.h	
Frame DMA API	414
vtss_api/include/ vtss_gfp_api.h	
GFP API	432
vtss_api/include/ vtss_hqos_api.h	
HQoS API	432
vtss_api/include/ vtss_i2c_api.h	
I2C API	432
vtss_api/include/ vtss_init_api.h	
Initialization API	433
vtss_api/include/ vtss_l2_api.h	
Layer 2 API	446
vtss_api/include/ vtss_l3_api.h	
L3 routing API	504
vtss_api/include/ vtss_mac10g_api.h	
MAC10G API	504
vtss_api/include/ vtss_macsec_api.h	
??	
vtss_api/include/ vtss_misc_api.h	
Miscellaneous API	504
vtss_api/include/ vtss_mpls_api.h	
MPLS API	539
vtss_api/include/ vtss_oam_api.h	
OAM API	539
vtss_api/include/ vtss_oha_api.h	
OHA API	539

vtss_api/include/vtss_os.h	
OS Layer API	540
vtss_api/include/vtss_os_custom.h	
OS custom header file	540
vtss_api/include/vtss_os_ecos.h	
ECos OS API	545
vtss_api/include/vtss_os_linux.h	
Linux OS API	555
vtss_api/include/vtss_otn_api.h	
OTN API	562
vtss_api/include/vtss_packet_api.h	
Packet API	562
vtss_api/include/vtss_pcs_10gbase_r_api.h	
PCS_10BASE_R API	584
vtss_api/include/vtss_phy_10g_api.h	
10G PHY API	584
vtss_api/include/vtss_phy_api.h	
PHY API	584
vtss_api/include/vtss_phy_ts_api.h	
PHY TimeStamping API	661
vtss_api/include/vtss_port_api.h	
Port API	662
vtss_api/include/vtss_qos_api.h	
QoS API	674
vtss_api/include/vtss_rab_api.h	
RAB API	683
vtss_api/include/vtss_security_api.h	
Security API	683
vtss_api/include/vtss_sfi4_api.h	
SFI4 API	693
vtss_api/include/vtss_sync_api.h	
Synchronization API	694
vtss_api/include/vtss_tfi5_api.h	
TFI5 API	697
vtss_api/include/vtss_ts_api.h	
TimeStamping API	698
vtss_api/include/vtss_upi_api.h	
Define UPI API interface	714
vtss_api/include/vtss_wis_api.h	
EWIS layer API	714
vtss_api/include/vtss_xaui_api.h	
XAUI API	714
vtss_api/include/vtss_xfi_api.h	
XFI API	715
vtss_api/include/vtss/api/l2_types.h	
Layer 2 Public API Header for l2	333
vtss_api/include/vtss/api/options.h	
Features and options	334
vtss_api/include/vtss/api/phy.h	
PHY Public API Header	351
vtss_api/include/vtss/api/port.h	
Port Public API Header	353
vtss_api/include/vtss/api/types.h	
Generic types API	366

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`
- `vtss_phy_power_mode_t power_mode`
- `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 power_mode

`vtss_phy_power_mode_t` `port_custom_conf_t::power_mode`

PHY power mode

Definition at line 281 of file port.h.

6.1.2.10 exc_col_cont

`BOOL` `port_custom_conf_t::exc_col_cont`

Excessive collision continuation

Definition at line 283 of file port.h.

6.1.2.11 adv_dis

`u8 port_custom_conf_t::adv_dis`

Auto neg advertisement disable

Definition at line 284 of file port.h.

6.1.2.12 max_tags

`u8 port_custom_conf_t::max_tags`

Maximum number of tags

Definition at line 285 of file port.h.

6.1.2.13 oper_up

`BOOL port_custom_conf_t::oper_up`

Force operational state up

Definition at line 286 of file port.h.

6.1.2.14 frame_length_chk

`BOOL port_custom_conf_t::frame_length_chk`

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

Definition at line 287 of file port.h.

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

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

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 tag_vtss_fdma_list Struct Reference

Software DCB structure.

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

Data Fields

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

6.3.1 Detailed Description

Software DCB structure.

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

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

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

Definition at line 346 of file vtss_fdma_api.h.

6.3.2 Field Documentation

6.3.2.1 frm_ptr

`u8* tag_vtss_fdma_list::frm_ptr`

XTR:

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

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

INJ/AFI:

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

Definition at line 415 of file vtss_fdma_api.h.

6.3.2.2 act_len

```
u32 tag_vtss_fdma_list::act_len
```

XTR:

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

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

Definition at line 441 of file vtss_fdma_api.h.

6.3.2.3 alloc_ptr

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

XTR:

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

Not used.

Definition at line 454 of file vtss_fdma_api.h.

6.3.2.4 rx_info

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

XTR:

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

Not used.

Definition at line 465 of file vtss_fdma_api.h.

6.3.2.5 sw_tstamp

```
VTSS_OS_TIMESTAMP_TYPE tag_vtss_fdma_list::sw_tstamp
```

XTR:

Unused. In V3+, it's part of the [vtss_packet_rx_info_t](#) structure and is called sw_tstamp. **INJ:**

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

. AFI:

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

Definition at line 508 of file vtss_fdma_api.h.

6.3.2.6 user

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

XTR/INJ/AFI:

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

Definition at line 515 of file vtss_fdma_api.h.

6.3.2.7 afi_frm_cnt

```
u64 tag_vtss_fdma_list::afi_frm_cnt
```

XTR/INJ

Not used. **AFI**

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

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

Definition at line 683 of file vtss_fdma_api.h.

6.3.2.8 afi_seq_number

```
u32 tag_vtss_fdma_list::afi_seq_number
```

XTR/INJ

Not used. **AFI**

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

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

Definition at line 700 of file vtss_fdma_api.h.

6.3.2.9 next

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

XTR:

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

INJ:

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

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

Definition at line 716 of file vtss_fdma_api.h.

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

- vtss_api/include/vtss_fdma_api.h

6.4 vtss_ace_frame_arp_t Struct Reference

Frame data for VTSS_ACE_TYPE_ARP.

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

Data Fields

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

6.4.1 Detailed Description

Frame data for VTSS_ACE_TYPE_ARP.

Definition at line 450 of file vtss_security_api.h.

6.4.2 Field Documentation

6.4.2.1 smac

```
vtss_ace_u48_t vtss_ace_frame_arp_t::smac
```

SMAC

Definition at line 452 of file vtss_security_api.h.

6.4.2.2 arp

```
vtss_ace_bit_t vtss_ace_frame_arp_t::arp
```

Opcode ARP/RARP

Definition at line 453 of file vtss_security_api.h.

6.4.2.3 req

`vtss_ace_bit_t vtss_ace_frame_arp_t::req`

Opcode request/reply

Definition at line 454 of file vtss_security_api.h.

6.4.2.4 unknown

`vtss_ace_bit_t vtss_ace_frame_arp_t::unknown`

Opcode unknown

Definition at line 455 of file vtss_security_api.h.

6.4.2.5 smac_match

`vtss_ace_bit_t vtss_ace_frame_arp_t::smac_match`

Sender MAC matches SMAC

Definition at line 456 of file vtss_security_api.h.

6.4.2.6 dmac_match

`vtss_ace_bit_t vtss_ace_frame_arp_t::dmac_match`

Target MAC matches DMAC

Definition at line 457 of file vtss_security_api.h.

6.4.2.7 length

`vtss_ace_bit_t vtss_ace_frame_arp_t::length`

Protocol addr. length 4, hardware length 6

Definition at line 458 of file vtss_security_api.h.

6.4.2.8 ip

`vtss_ace_bit_t` `vtss_ace_frame_arp_t::ip`

Protocol address type IP

Definition at line 459 of file `vtss_security_api.h`.

6.4.2.9 ethernet

`vtss_ace_bit_t` `vtss_ace_frame_arp_t::ethernet`

Hardware address type Ethernet

Definition at line 460 of file `vtss_security_api.h`.

6.4.2.10 sip

`vtss_ace_ip_t` `vtss_ace_frame_arp_t::sip`

Sender IP address

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

6.4.2.11 dip

`vtss_ace_ip_t` `vtss_ace_frame_arp_t::dip`

Target IP address

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

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

- `vtss_api/include/vtss_security_api.h`

6.5 vtss_ace_frame_etype_t Struct Reference

Frame data for VTSS_ACE_TYPEETYPE.

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

Data Fields

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

6.5.1 Detailed Description

Frame data for VTSS_ACE_TYPE_ETYPE.

Definition at line 422 of file vtss_security_api.h.

6.5.2 Field Documentation

6.5.2.1 dmac

`vtss_ace_u48_t vtss_ace_frame_etype_t::dmac`

DMAC

Definition at line 424 of file vtss_security_api.h.

6.5.2.2 smac

`vtss_ace_u48_t vtss_ace_frame_etype_t::smac`

SMAC

Definition at line 425 of file vtss_security_api.h.

6.5.2.3 etype

`vtss_ace_u16_t vtss_ace_frame_etype_t::etype`

Ethernet Type value

Definition at line 426 of file vtss_security_api.h.

6.5.2.4 data

```
vtss_ace_u16_t vtss_ace_frame_etype_t::data
```

MAC data

Definition at line 427 of file vtss_security_api.h.

6.5.2.5 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

6.6 vtss_ace_frame_ipv4_t Struct Reference

Frame data for VTSS_ACE_TYPE_IPV4.

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

Data Fields

- vtss_ace_bit_t ttl
- vtss_ace_bit_t fragment
- vtss_ace_bit_t options
- vtss_ace_u8_t ds
- vtss_ace_u8_t proto
- vtss_ace_ip_t sip
- vtss_ace_ip_t dip
- vtss_ace_u48_t data
- vtss_ace_udp_tcp_t sport
- vtss_ace_udp_tcp_t dport
- vtss_ace_bit_t tcp_fin
- vtss_ace_bit_t tcp_syn
- vtss_ace_bit_t tcp_RST
- vtss_ace_bit_t tcp_psh
- vtss_ace_bit_t tcp_ack
- vtss_ace_bit_t tcp_urg
- vtss_ace_bit_t sip_eq_dip
- vtss_ace_bit_t sport_eq_dport
- vtss_ace_bit_t seq_zero
- vtss_ace_ptp_t ptp
- vtss_ace_sip_smac_t sip_smac

6.6.1 Detailed Description

Frame data for VTSS_ACE_TYPE_IPV4.

Definition at line 466 of file vtss_security_api.h.

6.6.2 Field Documentation

6.6.2.1 ttl

`vtss_ace_bit_t` `vtss_ace_frame_ipv4_t::ttl`

TTL zero

Definition at line 468 of file vtss_security_api.h.

6.6.2.2 fragment

`vtss_ace_bit_t` `vtss_ace_frame_ipv4_t::fragment`

Fragment

Definition at line 469 of file vtss_security_api.h.

6.6.2.3 options

`vtss_ace_bit_t` `vtss_ace_frame_ipv4_t::options`

Header options

Definition at line 470 of file vtss_security_api.h.

6.6.2.4 ds

`vtss_ace_u8_t` `vtss_ace_frame_ipv4_t::ds`

DS field

Definition at line 471 of file vtss_security_api.h.

6.6.2.5 proto

`vtss_ace_u8_t` `vtss_ace_frame_ipv4_t::proto`

Protocol

Definition at line 472 of file vtss_security_api.h.

6.6.2.6 sip

`vtss_ace_ip_t` `vtss_ace_frame_ipv4_t::sip`

Source IP address

Definition at line 473 of file vtss_security_api.h.

6.6.2.7 dip

`vtss_ace_ip_t` `vtss_ace_frame_ipv4_t::dip`

Destination IP address

Definition at line 474 of file vtss_security_api.h.

6.6.2.8 data

`vtss_ace_u48_t` `vtss_ace_frame_ipv4_t::data`

Not UDP/TCP: IP data

Definition at line 475 of file vtss_security_api.h.

6.6.2.9 sport

`vtss_ace_udp_tcp_t` `vtss_ace_frame_ipv4_t::sport`

UDP/TCP: Source port

Definition at line 476 of file vtss_security_api.h.

6.6.2.10 dport

`vtss_ace_udp_tcp_t` `vtss_ace_frame_ipv4_t::dport`

UDP/TCP: Destination port

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

6.6.2.11 tcp_fin

`vtss_ace_bit_t` `vtss_ace_frame_ipv4_t::tcp_fin`

TCP FIN

Definition at line 478 of file `vtss_security_api.h`.

6.6.2.12 tcp_syn

`vtss_ace_bit_t` `vtss_ace_frame_ipv4_t::tcp_syn`

TCP SYN

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

6.6.2.13 tcp_RST

`vtss_ace_bit_t` `vtss_ace_frame_ipv4_t::tcp_RST`

TCP RST

Definition at line 480 of file `vtss_security_api.h`.

6.6.2.14 tcp_psh

`vtss_ace_bit_t` `vtss_ace_frame_ipv4_t::tcp_psh`

TCP PSH

Definition at line 481 of file `vtss_security_api.h`.

6.6.2.15 tcp_ack

`vtss_ace_bit_t` vtss_ace_frame_ipv4_t::tcp_ack

TCP ACK

Definition at line 482 of file vtss_security_api.h.

6.6.2.16 tcp_urg

`vtss_ace_bit_t` vtss_ace_frame_ipv4_t::tcp_urg

TCP URG

Definition at line 483 of file vtss_security_api.h.

6.6.2.17 sip_eq_dip

`vtss_ace_bit_t` vtss_ace_frame_ipv4_t::sip_eq_dip

SIP equals DIP

Definition at line 484 of file vtss_security_api.h.

6.6.2.18 sport_eq_dport

`vtss_ace_bit_t` vtss_ace_frame_ipv4_t::sport_eq_dport

SPORT equals DPORt

Definition at line 485 of file vtss_security_api.h.

6.6.2.19 seq_zero

`vtss_ace_bit_t` vtss_ace_frame_ipv4_t::seq_zero

TCP sequence number is zero

Definition at line 486 of file vtss_security_api.h.

6.6.2.20 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`.

6.6.2.21 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`

6.7 `vtss_ace_frame_ipv6_t` Struct Reference

Frame data for `VTSS_ACE_TYPE_IPV6`.

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

Data Fields

- `vtss_ace_u8_t proto`
- `vtss_ace_u128_t sip`
- `vtss_ace_bit_t ttl`
- `vtss_ace_u8_t ds`
- `vtss_ace_u48_t data`
- `vtss_ace_udp_tcp_t sport`
- `vtss_ace_udp_tcp_t dport`
- `vtss_ace_bit_t tcp_fin`
- `vtss_ace_bit_t tcp_syn`
- `vtss_ace_bit_t tcp_RST`
- `vtss_ace_bit_t tcp_psh`
- `vtss_ace_bit_t tcp_ack`
- `vtss_ace_bit_t tcp_urg`
- `vtss_ace_bit_t sip_eq_dip`
- `vtss_ace_bit_t sport_eq_dport`
- `vtss_ace_bit_t seq_zero`
- `vtss_ace_ptp_t ptp`

6.7.1 Detailed Description

Frame data for VTSS_ACE_TYPE_IPV6.

Definition at line 494 of file vtss_security_api.h.

6.7.2 Field Documentation

6.7.2.1 proto

`vtss_ace_u8_t` vtss_ace_frame_ipv6_t::proto

IPv6 protocol

Definition at line 496 of file vtss_security_api.h.

6.7.2.2 sip

`vtss_ace_u128_t` vtss_ace_frame_ipv6_t::sip

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

Definition at line 497 of file vtss_security_api.h.

6.7.2.3 ttl

`vtss_ace_bit_t` vtss_ace_frame_ipv6_t::ttl

TTL zero

Definition at line 498 of file vtss_security_api.h.

6.7.2.4 ds

`vtss_ace_u8_t` vtss_ace_frame_ipv6_t::ds

DS field

Definition at line 499 of file vtss_security_api.h.

6.7.2.5 data

`vtss_ace_u48_t vtss_ace_frame_ipv6_t::data`

Not UDP/TCP: IP data

Definition at line 500 of file `vtss_security_api.h`.

6.7.2.6 sport

`vtss_ace_udp_tcp_t vtss_ace_frame_ipv6_t::sport`

UDP/TCP: Source port

Definition at line 501 of file `vtss_security_api.h`.

6.7.2.7 dport

`vtss_ace_udp_tcp_t vtss_ace_frame_ipv6_t::dport`

UDP/TCP: Destination port

Definition at line 502 of file `vtss_security_api.h`.

6.7.2.8 tcp_fin

`vtss_ace_bit_t vtss_ace_frame_ipv6_t::tcp_fin`

TCP FIN

Definition at line 503 of file `vtss_security_api.h`.

6.7.2.9 tcp_syn

`vtss_ace_bit_t vtss_ace_frame_ipv6_t::tcp_syn`

TCP SYN

Definition at line 504 of file `vtss_security_api.h`.

6.7.2.10 tcp_rst

`vtss_ace_bit_t` `vtss_ace_frame_ipv6_t::tcp_rst`

TCP RST

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

6.7.2.11 tcp_psh

`vtss_ace_bit_t` `vtss_ace_frame_ipv6_t::tcp_psh`

TCP PSH

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

6.7.2.12 tcp_ack

`vtss_ace_bit_t` `vtss_ace_frame_ipv6_t::tcp_ack`

TCP ACK

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

6.7.2.13 tcp_urg

`vtss_ace_bit_t` `vtss_ace_frame_ipv6_t::tcp_urg`

TCP URG

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

6.7.2.14 sip_eq_dip

`vtss_ace_bit_t` `vtss_ace_frame_ipv6_t::sip_eq_dip`

SIP equals DIP

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

6.7.2.15 sport_eq_dport

`vtss_ace_bit_t vtss_ace_frame_ipv6_t::sport_eq_dport`

SPORT equals DPORt

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

6.7.2.16 seq_zero

`vtss_ace_bit_t vtss_ace_frame_ipv6_t::seq_zero`

TCP sequence number is zero

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

6.7.2.17 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`

6.8 vtss_ace_frame_llc_t Struct Reference

Frame data for VTSS_ACE_TYPE LLC.

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

Data Fields

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

6.8.1 Detailed Description

Frame data for VTSS_ACE_TYPE LLC.

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

6.8.2 Field Documentation

6.8.2.1 dmac

`vtss_ace_u48_t vtss_ace_frame_llc_t::dmac`

DMAC

Definition at line 436 of file vtss_security_api.h.

6.8.2.2 smac

`vtss_ace_u48_t vtss_ace_frame_llc_t::smac`

SMAC

Definition at line 437 of file vtss_security_api.h.

6.8.2.3 llc

`vtss_ace_u32_t vtss_ace_frame_llc_t::llc`

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

Definition at line 438 of file vtss_security_api.h.

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

- `vtss_api/include/vtss_security_api.h`

6.9 vtss_ace_frame_snap_t Struct Reference

Frame data for VTSS_ACE_TYPE_SNAP.

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

Data Fields

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

6.9.1 Detailed Description

Frame data for VTSS_ACE_TYPE_SNAP.

Definition at line 442 of file vtss_security_api.h.

6.9.2 Field Documentation

6.9.2.1 dmac

`vtss_ace_u48_t vtss_ace_frame_snap_t::dmac`

DMAC

Definition at line 444 of file vtss_security_api.h.

6.9.2.2 smac

`vtss_ace_u48_t vtss_ace_frame_snap_t::smac`

SMAC

Definition at line 445 of file vtss_security_api.h.

6.9.2.3 snap

`vtss_ace_u40_t vtss_ace_frame_snap_t::snap`

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

Definition at line 446 of file vtss_security_api.h.

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

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

6.10 vtss_ace_ptp_t Struct Reference

PTP header filtering.

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

Data Fields

- `BOOL enable`
- `vtss_ace_u32_t header`

6.10.1 Detailed Description

PTP header filtering.

Definition at line 391 of file vtss_security_api.h.

6.10.2 Field Documentation

6.10.2.1 enable

`BOOL vtss_ace_ptp_t::enable`

Enable PTP header filtering

Definition at line 393 of file vtss_security_api.h.

6.10.2.2 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`

6.11 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`

6.11.1 Detailed Description

SIP/SMAC filtering.

Definition at line 398 of file vtss_security_api.h.

6.11.2 Field Documentation

6.11.2.1 enable

`BOOL vtss_ace_sip_smac_t::enable`

Enable SIP/SMAC filtering

Definition at line 400 of file vtss_security_api.h.

6.11.2.2 sip

`vtss_ip_t vtss_ace_sip_smac_t::sip`

SIP

Definition at line 401 of file vtss_security_api.h.

6.11.2.3 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](#)

6.12 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`}

6.12.1 Detailed Description

Access Control Entry.

Definition at line 518 of file vtss_security_api.h.

6.12.2 Field Documentation

6.12.2.1 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.

6.12.2.2 port_list

`BOOL vtss_ace_t::port_list [VTSS_PORT_ARRAY_SIZE]`

Port list

Definition at line 530 of file vtss_security_api.h.

6.12.2.3 policy

`vtss_ace_u8_t` `vtss_ace_t::policy`

Policy number

Definition at line 532 of file vtss_security_api.h.

6.12.2.4 type

`vtss_ace_type_t` `vtss_ace_t::type`

ACE frame type

Definition at line 533 of file vtss_security_api.h.

6.12.2.5 action

`vtss_acl_action_t` `vtss_ace_t::action`

ACE action

Definition at line 534 of file vtss_security_api.h.

6.12.2.6 dmac_mc

`vtss_ace_bit_t` `vtss_ace_t::dmac_mc`

Multicast DMAC

Definition at line 536 of file vtss_security_api.h.

6.12.2.7 dmac_bc

`vtss_ace_bit_t` `vtss_ace_t::dmac_bc`

Broadcast DMAC

Definition at line 537 of file vtss_security_api.h.

6.12.2.8 vlan

`vtss_ace_vlan_t` `vtss_ace_t::vlan`

VLAN Tag

Definition at line 539 of file vtss_security_api.h.

6.12.2.9 etype

`vtss_ace_frame_etype_t` `vtss_ace_t::etype`

VTSS_ACE_TYPE_ETYPE

Definition at line 544 of file vtss_security_api.h.

6.12.2.10 llc

`vtss_ace_frame_llc_t` `vtss_ace_t::llc`

VTSS_ACE_TYPE_LLCC

Definition at line 545 of file vtss_security_api.h.

6.12.2.11 snap

`vtss_ace_frame_snap_t` `vtss_ace_t::snap`

VTSS_ACE_TYPE_SNAP

Definition at line 546 of file vtss_security_api.h.

6.12.2.12 arp

`vtss_ace_frame_arp_t` `vtss_ace_t::arp`

VTSS_ACE_TYPE_ARP

Definition at line 547 of file vtss_security_api.h.

6.12.2.13 ipv4

`vtss_ace_frame_ipv4_t vtss_ace_t::ipv4`

VTSS_ACE_TYPE_IPV4

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

6.12.2.14 ipv6

`vtss_ace_frame_ipv6_t vtss_ace_t::ipv6`

VTSS_ACE_TYPE_IPV6

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

6.12.2.15 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`

6.13 `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`

6.13.1 Detailed Description

ACE VLAN information.

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

6.13.2 Field Documentation

6.13.2.1 vid

`vtss_ace_vid_t` `vtss_ace_vlan_t::vid`

VLAN ID (12 bit)

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

6.13.2.2 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`.

6.13.2.3 cfi

`vtss_ace_bit_t` `vtss_ace_vlan_t::cfi`

CFI/DEI

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

6.13.2.4 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`

6.14 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`

6.14.1 Detailed Description

ACL Action.

Definition at line 238 of file vtss_security_api.h.

6.14.2 Field Documentation

6.14.2.1 `cpu`

```
BOOL vtss_acl_action_t::cpu
```

Forward to CPU

Definition at line 240 of file vtss_security_api.h.

6.14.2.2 `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.

6.14.2.3 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.

6.14.2.4 police

`BOOL` `vtss_acl_action_t::police`

Enable policer

Definition at line 243 of file vtss_security_api.h.

6.14.2.5 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.

6.14.2.6 evc_police

`BOOL` `vtss_acl_action_t::evc_police`

Enable EVC policer

Definition at line 247 of file vtss_security_api.h.

6.14.2.7 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.

6.14.2.8 learn

`BOOL vtss_acl_action_t::learn`

Allow learning

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

6.14.2.9 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`.

6.14.2.10 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`.

6.14.2.11 mirror

`BOOL vtss_acl_action_t::mirror`

Enable mirroring

Definition at line 260 of file `vtss_security_api.h`.

6.14.2.12 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`

6.15 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`

6.15.1 Detailed Description

ACL policer configuration.

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

6.15.2 Field Documentation

6.15.2.1 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`.

6.15.2.2 bit_rate

```
vtss_bitrate_t vtss_acl_policer_conf_t::bit_rate
```

Bit rate

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

6.15.2.3 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`

6.16 `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`

6.16.1 Detailed Description

ACL port configuration.

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

6.16.2 Field Documentation

6.16.2.1 `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`.

6.16.2.2 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](#)

6.17 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](#)

6.17.1 Detailed Description

Aggregation traffic distribution mode.

Definition at line 39 of file l2_types.h.

6.17.2 Field Documentation

6.17.2.1 smac_enable

```
BOOL vtss_aggr_mode_t::smac_enable
```

Source MAC address

Definition at line 41 of file l2_types.h.

6.17.2.2 dmac_enable

`BOOL vtss_aggr_mode_t::dmac_enable`

Destination MAC address

Definition at line 42 of file l2_types.h.

6.17.2.3 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.

6.17.2.4 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](#)

6.18 vtss_aneg_t Struct Reference

Auto negotiation struct.

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

Data Fields

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

6.18.1 Detailed Description

Auto negotiation struct.

Definition at line 482 of file types.h.

6.18.2 Field Documentation

6.18.2.1 obey_pause

`BOOL vtss_aneg_t::obey_pause`

This port should obey PAUSE frames

Definition at line 484 of file types.h.

6.18.2.2 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`

6.19 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`

6.19.1 Detailed Description

API lock structure.

Definition at line 285 of file vtss_misc_api.h.

6.19.2 Field Documentation

6.19.2.1 inst

`vtss_inst_t vtss_api_lock_t::inst`

Target instance reference

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

6.19.2.2 function

`const char* vtss_api_lock_t::function`

Function name

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

6.19.2.3 file

`const char* vtss_api_lock_t::file`

File name

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

6.19.2.4 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`

6.20 vtss_basic_counters_t Struct Reference

Basic counters structure.

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

Data Fields

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

6.20.1 Detailed Description

Basic counters structure.

Definition at line 377 of file vtss_port_api.h.

6.20.2 Field Documentation

6.20.2.1 rx_frames

```
u32 vtss_basic_counters_t::rx_frames
```

Rx frames

Definition at line 379 of file vtss_port_api.h.

6.20.2.2 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](#)

6.21 vtss_chip_id_t Struct Reference

Chip ID.

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

Data Fields

- `u16 part_number`
- `u16 revision`

6.21.1 Detailed Description

Chip ID.

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

6.21.2 Field Documentation

6.21.2.1 part_number

`u16 vtss_chip_id_t::part_number`

BCD encoded part number

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

6.21.2.2 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`

6.22 vtss_counter_pair_t Struct Reference

Counter pair.

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

Data Fields

- `vtss_counter_t frames`
- `vtss_counter_t bytes`

6.22.1 Detailed Description

Counter pair.

Definition at line 1111 of file types.h.

6.22.2 Field Documentation

6.22.2.1 frames

`vtss_counter_t vtss_counter_pair_t::frames`

Number of frames

Definition at line 1112 of file types.h.

6.22.2.2 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`

6.23 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`

6.23.1 Detailed Description

Debug information structure.

Definition at line 243 of file vtss_misc_api.h.

6.23.2 Field Documentation

6.23.2.1 layer

```
vtss_debug_layer_t vtss_debug_info_t::layer
```

Layer

Definition at line 244 of file vtss_misc_api.h.

6.23.2.2 group

```
vtss_debug_group_t vtss_debug_info_t::group
```

Function group

Definition at line 245 of file vtss_misc_api.h.

6.23.2.3 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.

6.23.2.4 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.

6.23.2.5 full

```
BOOL vtss_debug_info_t::full
```

Full information dump

Definition at line 248 of file vtss_misc_api.h.

6.23.2.6 clear

```
BOOL vtss_debug_info_t::clear
```

Clear counters

Definition at line 249 of file vtss_misc_api.h.

6.23.2.7 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](#)

6.24 vtss_debug_lock_t Struct Reference

API debug lock structure.

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

Data Fields

- [vtss_chip_no_t chip_no](#)

6.24.1 Detailed Description

API debug lock structure.

Definition at line 307 of file vtss_misc_api.h.

6.24.2 Field Documentation

6.24.2.1 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`

6.25 `vtss_dgroup_port_conf_t` Struct Reference

Destination group port configuration.

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

Data Fields

- `vtss_dgroup_no_t dgroup_no`

6.25.1 Detailed Description

Destination group port configuration.

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

6.25.2 Field Documentation

6.25.2.1 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`

6.26 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`

6.26.1 Detailed Description

Dual leaky buckets policer configuration.

Definition at line 237 of file vtss_qos_api.h.

6.26.2 Field Documentation

6.26.2.1 type

```
vtss_policer_type_t vtss_dlb_policer_conf_t::type
```

Policer type

Definition at line 238 of file vtss_qos_api.h.

6.26.2.2 enable

```
BOOL vtss_dlb_policer_conf_t::enable
```

Enable/disable policer

Definition at line 239 of file vtss_qos_api.h.

6.26.2.3 cf

`BOOL vtss_dlb_policer_conf_t::cf`

Coupling Flag

Definition at line 243 of file vtss_qos_api.h.

6.26.2.4 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.

6.26.2.5 cir

`vtss_bitrate_t vtss_dlb_policer_conf_t::cir`

Committed Information Rate

Definition at line 245 of file vtss_qos_api.h.

6.26.2.6 cbs

`vtss_burst_level_t vtss_dlb_policer_conf_t::cbs`

Committed Burst Size

Definition at line 246 of file vtss_qos_api.h.

6.26.2.7 eir

`vtss_bitrate_t vtss_dlb_policer_conf_t::eir`

Excess Information Rate

Definition at line 247 of file vtss_qos_api.h.

6.26.2.8 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`

6.27 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`

6.27.1 Detailed Description

ECE action.

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

6.27.2 Field Documentation

6.27.2.1 dir

`vtss_ece_dir_t` `vtss_ece_action_t::dir`

Traffic direction

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

6.27.2.2 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.

6.27.2.3 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.

6.27.2.4 evc_id

`vtss_evc_id_t` `vtss_ece_action_t::evc_id`

EVC ID

Definition at line 569 of file vtss_evc_api.h.

6.27.2.5 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.

6.27.2.6 prio_enable

`BOOL` `vtss_ece_action_t::prio_enable`

Enable priority classification

Definition at line 572 of file vtss_evc_api.h.

6.27.2.7 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](#)

6.28 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](#)

6.28.1 Detailed Description

ECE IPv4 information.

Definition at line 461 of file vtss_evc_api.h.

6.28.2 Field Documentation

6.28.2.1 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.

6.28.2.2 fragment

`vtss_vcap_bit_t` `vtss_ece_frame_ipv4_t::fragment`

Fragment

Definition at line 464 of file `vtss_evc_api.h`.

6.28.2.3 proto

`vtss_vcap_u8_t` `vtss_ece_frame_ipv4_t::proto`

Protocol

Definition at line 465 of file `vtss_evc_api.h`.

6.28.2.4 sip

`vtss_vcap_ip_t` `vtss_ece_frame_ipv4_t::sip`

Source IP address

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

6.28.2.5 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`.

6.28.2.6 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`

6.29 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`

6.29.1 Detailed Description

ECE IPv6 information.

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

6.29.2 Field Documentation

6.29.2.1 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`.

6.29.2.2 proto

```
vtss_vcap_u8_t vtss_ece_frame_ipv6_t::proto
```

Protocol

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

6.29.2.3 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`.

6.29.2.4 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`.

6.29.2.5 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`

6.30 vtss_ece_key_t Struct Reference

ECE key.

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

Data Fields

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

6.30.1 Detailed Description

ECE key.

Definition at line 490 of file vtss_evc_api.h.

6.30.2 Field Documentation

6.30.2.1 port_list

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

UNI port list

Definition at line 492 of file vtss_evc_api.h.

6.30.2.2 mac

`vtss_ece_mac_t` `vtss_ece_key_t::mac`

MAC header

Definition at line 493 of file vtss_evc_api.h.

6.30.2.3 tag

`vtss_ece_tag_t` `vtss_ece_key_t::tag`

Tag

Definition at line 494 of file vtss_evc_api.h.

6.30.2.4 type

`vtss_ece_type_t` `vtss_ece_key_t::type`

Frame type

Definition at line 498 of file vtss_evc_api.h.

6.30.2.5 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`.

6.30.2.6 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`.

6.30.2.7 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`

6.31 `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`

6.31.1 Detailed Description

ECE MAC information.

Definition at line 420 of file `vtss_evc_api.h`.

6.31.2 Field Documentation

6.31.2.1 dmac_mc

`vtss_vcap_bit_t` vtss_ece_mac_t::dmac_mc

Multicast DMAC

Definition at line 426 of file vtss_evc_api.h.

6.31.2.2 dmac_bc

`vtss_vcap_bit_t` vtss_ece_mac_t::dmac_bc

Broadcast DMAC

Definition at line 427 of file vtss_evc_api.h.

6.31.2.3 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](#)

6.32 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`

6.32.1 Detailed Description

ECE outer tag.

Definition at line 517 of file vtss_evc_api.h.

6.32.2 Field Documentation

6.32.2.1 enable

```
BOOL vtss_ece_outer_tag_t::enable
```

Enable tag (VTSS_ECE_DIR_NNI_TO_UNI only)

Definition at line 519 of file vtss_evc_api.h.

6.32.2.2 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.

6.32.2.3 pcp

```
vtss_tagprio_t vtss_ece_outer_tag_t::pcp
```

PCP value

Definition at line 529 of file vtss_evc_api.h.

6.32.2.4 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](#)

6.33 vtss_ece_t Struct Reference

EVC Control Entry.

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

Data Fields

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

6.33.1 Detailed Description

EVC Control Entry.

Definition at line 582 of file vtss_evc_api.h.

6.33.2 Field Documentation

6.33.2.1 id

```
vtss_ece_id_t vtss_ece_t::id
```

Entry ID

Definition at line 583 of file vtss_evc_api.h.

6.33.2.2 key

```
vtss_ece_key_t vtss_ece_t::key
```

ECE key

Definition at line 584 of file vtss_evc_api.h.

6.33.2.3 action

`vtss_ece_action_t` `vtss_ece_t::action`

ECE action

Definition at line 585 of file `vtss_evc_api.h`.

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

- `vtss_api/include/vtss_evc_api.h`

6.34 `vtss_ece_tag_t` Struct Reference

ECE tag information.

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

Data Fields

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

6.34.1 Detailed Description

ECE tag information.

Definition at line 433 of file `vtss_evc_api.h`.

6.34.2 Field Documentation

6.34.2.1 vid

`vtss_vcap_vr_t` `vtss_ece_tag_t::vid`

VLAN ID (12 bit)

Definition at line 435 of file `vtss_evc_api.h`.

6.34.2.2 pcp

`vtss_vcap_u8_t` `vtss_ece_tag_t::pcp`

PCP (3 bit)

Definition at line 436 of file vtss_evc_api.h.

6.34.2.3 dei

`vtss_vcap_bit_t` `vtss_ece_tag_t::dei`

DEI

Definition at line 437 of file vtss_evc_api.h.

6.34.2.4 tagged

`vtss_vcap_bit_t` `vtss_ece_tag_t::tagged`

Tagged/untagged frame

Definition at line 438 of file vtss_evc_api.h.

6.34.2.5 s_tagged

`vtss_vcap_bit_t` `vtss_ece_tag_t::s_tagged`

S-tagged/C-tagged frame

Definition at line 439 of file vtss_evc_api.h.

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

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

6.35 vtss_eee_port_conf_t Struct Reference

EEE port configuration.

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

Data Fields

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

6.35.1 Detailed Description

EEE port configuration.

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

6.35.2 Field Documentation

6.35.2.1 `eee_ena`

```
BOOL vtss_eee_port_conf_t::eee_ena
```

Enable EEE

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

6.35.2.2 `eee_fast_queues`

```
u8 vtss_eee_port_conf_t::eee_fast_queues
```

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

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

6.35.2.3 `tx_tw`

```
u16 vtss_eee_port_conf_t::tx_tw
```

Time from path is activated until frame transmission restarted.

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

6.35.2.4 lp_advertisement

```
u8 vtss_eee_port_conf_t::lp_advertisement
```

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

Definition at line 1224 of file vtss_misc_api.h.

6.35.2.5 optimized_for_power

```
BOOL vtss_eee_port_conf_t::optimized_for_power
```

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

Definition at line 1226 of file vtss_misc_api.h.

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

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

6.36 vtss_eee_port_counter_t Struct Reference

EEE port counters (JR only)

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

Data Fields

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

6.36.1 Detailed Description

EEE port counters (JR only)

Definition at line 1246 of file vtss_misc_api.h.

6.36.2 Field Documentation

6.36.2.1 fill_level_get

```
BOOL vtss_eee_port_counter_t::fill_level_get
```

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

Definition at line 1248 of file vtss_misc_api.h.

6.36.2.2 fill_level_thres

```
u32 vtss_eee_port_counter_t::fill_level_thres
```

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

Definition at line 1249 of file vtss_misc_api.h.

6.36.2.3 fill_level

```
u32 vtss_eee_port_counter_t::fill_level
```

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

Definition at line 1250 of file vtss_misc_api.h.

6.36.2.4 tx_out_bytes_get

```
BOOL vtss_eee_port_counter_t::tx_out_bytes_get
```

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

Definition at line 1251 of file vtss_misc_api.h.

6.36.2.5 tx_out_bytes

```
u32 vtss_eee_port_counter_t::tx_out_bytes
```

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

Definition at line 1252 of file vtss_misc_api.h.

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

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

6.37 vtss_eee_port_state_t Struct Reference

EEE port state (JR only)

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

Data Fields

- `vtss_eee_state_select_t select`
- `u32 val`

6.37.1 Detailed Description

EEE port state (JR only)

Definition at line 1239 of file vtss_misc_api.h.

6.37.2 Field Documentation

6.37.2.1 select

```
vtss_eee_state_select_t vtss_eee_port_state_t::select
```

State to change.

Definition at line 1241 of file vtss_misc_api.h.

6.37.2.2 val

```
u32 vtss_eee_port_state_t::val
```

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

Definition at line 1242 of file vtss_misc_api.h.

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

- vtss_api/include/vtss_misc_api.h

6.38 vtss_eps_port_conf_t Struct Reference

Port protection configuration.

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

Data Fields

- `vtss_eps_port_type_t type`
- `vtss_port_no_t port_no`

6.38.1 Detailed Description

Port protection configuration.

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

6.38.2 Field Documentation

6.38.2.1 type

`vtss_eps_port_type_t vtss_eps_port_conf_t::type`

Protection type

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

6.38.2.2 port_no

`vtss_port_no_t vtss_eps_port_conf_t::port_no`

Protection port or VTSS_PORT_NO_NONE

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

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

- `vtss_api/include/vtss_l2_api.h`

6.39 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`

6.39.1 Detailed Description

EVC configuration (excluding UNIs)

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

6.39.2 Field Documentation

6.39.2.1 learning

`BOOL vtss_evc_conf_t::learning`

Enable/disable learning

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

6.39.2.2 pb

`vtss_evc_pb_conf_t vtss_evc_conf_t::pb`

PB specific configuration

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

6.39.2.3 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`

6.40 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`

6.40.1 Detailed Description

EVC inner tag.

Definition at line 263 of file vtss_evc_api.h.

6.40.2 Field Documentation

6.40.2.1 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.

6.40.2.2 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.

6.40.2.3 vid

`vtss_vid_t` vtss_evc_inner_tag_t::vid

VLAN ID

Definition at line 267 of file vtss_evc_api.h.

6.40.2.4 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.

6.40.2.5 pcp

`vtss_tagprio_t` vtss_evc_inner_tag_t::pcp

PCP value

Definition at line 269 of file vtss_evc_api.h.

6.40.2.6 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](#)

6.41 vtss_evc_pb_conf_t Struct Reference

PB specific EVC configuration.

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

Data Fields

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

6.41.1 Detailed Description

PB specific EVC configuration.

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

6.41.2 Field Documentation

6.41.2.1 nni

`BOOL vtss_evc_pb_conf_t::nni [VTSS_PORT_ARRAY_SIZE]`

NNI configuration

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

6.41.2.2 ivid

`vtss_vid_t vtss_evc_pb_conf_t::ivid`

Internal VID

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

6.41.2.3 vid

`vtss_vid_t vtss_evc_pb_conf_t::vid`

NNI VID of outer tag

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

6.41.2.4 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.

6.41.2.5 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](#)

6.42 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](#)

6.42.1 Detailed Description

EVC port configuration.

Definition at line 150 of file vtss_evc_api.h.

6.42.2 Field Documentation

6.42.2.1 dei_colouring

`BOOL vtss_evc_port_conf_t::dei_colouring`

NNI: Enable colouring of DEI for received frames

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

6.42.2.2 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`.

6.42.2.3 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`

6.43 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`

6.43.1 Detailed Description

Fan specifications.

Definition at line 1148 of file vtss_misc_api.h.

6.43.2 Field Documentation

6.43.2.1 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.

6.43.2.2 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.

6.43.2.3 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.

6.43.2.4 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.

6.43.2.5 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](#)

6.44 vtss_fdma_cfg_t Struct Reference

FDMA configuration structure.

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

Data Fields

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

6.44.1 Detailed Description

FDMA configuration structure.

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

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

Definition at line 1820 of file vtss_fdma_api.h.

6.44.2 Field Documentation

6.44.2.1 enable

```
BOOL vtss_fdma_cfg_t::enable
```

Enable FDMA driver.

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

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

Definition at line 1830 of file vtss_fdma_api.h.

6.44.2.2 rx_mtu

```
u32 vtss_fdma_cfg_t::rx_mtu
```

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

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

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

Definition at line 1843 of file vtss_fdma_api.h.

6.44.2.3 rx_buf_cnt

```
u32 vtss_fdma_cfg_t::rx_buf_cnt
```

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

Definition at line 1852 of file vtss_fdma_api.h.

6.44.2.4 rx_alloc_cb

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

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

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

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

During initialization, the FDMA driver allocates [rx_buf_cnt](#) DCBs and checks whether the application will allocate the corresponding frame data, or it should do that itself. If [rx_alloc_cb\(\)](#) is NULL, the FDMA driver will call [VTSS_OS_MALLOC\(\)](#) to allocate the corresponding frame data. Otherwise it will call [rx_alloc_cb\(\)](#) to get it allocated.

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

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

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

Parameters

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

If - upon return from the [rx_alloc_cb\(\)](#) function, the DCB's [alloc_ptr](#) member is still NULL, one of two things will happen: i) If it happens during initialization ([vtss_fdma_cfg\(\)](#)), the function call will fail, and the FDMA will not be started. ii) If it happens at runtime ([rx_cb\(\)](#)/[vtss_fdma_dcb_release\(\)](#)), the DCB will be put in a special list that can only be released if the FDMA driver gets restarted.

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

Definition at line 1925 of file [vtss_fdma_api.h](#).

6.44.2.5 rx_dont_strip_vlan_tag

```
BOOL vtss_fdma_cfg_t::rx_dont_strip_vlan_tag
```

Don't strip VLAN tag.

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

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

Definition at line 1938 of file vtss_fdma_api.h.

6.44.2.6 rx_dont_reinsert_vlan_tag

```
BOOL vtss_fdma_cfg_t::rx_dont_reinsert_vlan_tag
```

Don't re-insert VLAN tag.

This is obsolete and is ignored by the FDMA code.

Definition at line 1945 of file vtss_fdma_api.h.

6.44.2.7 rx_allow_vlan_tag_mismatch

```
BOOL vtss_fdma_cfg_t::rx_allow_vlan_tag_mismatch
```

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

Definition at line 1952 of file vtss_fdma_api.h.

6.44.2.8 rx_allow_multiple_dcbs

```
BOOL vtss_fdma_cfg_t::rx_allow_multiple_dcbs
```

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

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

Definition at line 1964 of file vtss_fdma_api.h.

6.44.2.9 rx_cb

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

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

The function is invoked in the same context as the application invoked [vtss_fdma_irq_handler\(\)](#).

The parameters to the callback function are as follows:

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

Definition at line 1987 of file vtss_fdma_api.h.

6.44.2.10 tx_buf_cnt

```
u32 vtss_fdma_cfg_t::tx_buf_cnt
```

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

Definition at line 1993 of file vtss_fdma_api.h.

6.44.2.11 tx_done_cb

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

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

The function is invoked in the same context as the application invoked [vtss_fdma_irq_handler\(\)](#).

The parameters to the callback function are as follows:

1. ctxt: The ctxt passed to [vtss_fdma_irq_handler\(\)](#).
2. list: NULL-terminated list of software DCBs making up the frame. The last is always NULL-terminated.
3. rc: Return code. VTSS_RC_OK on success, otherwise failed to transmit.

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

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

Definition at line 2015 of file vtss_fdma_api.h.

6.44.2.12 afi_buf_cnt

```
u32 vtss_fdma_cfg_t::afi_buf_cnt
```

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

Definition at line 2022 of file vtss_fdma_api.h.

6.44.2.13 afi_done_cb

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

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

The function is invoked in the same context as the application invoked [vtss_fdma_irq_handler\(\)](#).

The parameters to the callback function are as follows:

1. ctxt: The ctxt passed to [vtss_fdma_irq_handler\(\)](#).
2. list: NULL-terminated list of software DCBs making up the frame. The last is always NULL-terminated.
3. rc: Return code. VTSS_RC_OK always.

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

Definition at line 2042 of file vtss_fdma_api.h.

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

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

6.45 vtss_fdma_ch_cfg_t Struct Reference

Channel configuration structure.

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

Data Fields

- [vtss_fdma_ch_usage_t](#) usage
- [vtss_packet_rx_grp_t](#) xtr_grp
- u32 inj_grp_mask
- [vtss_fdma_list_t](#) * list
- [vtss_fdma_list_t](#) *(* xtr_cb)(void *ctxt, [vtss_fdma_list_t](#) *list, [vtss_packet_rx_queue_t](#) qu)
- int prio
- [vtss_chip_no_t](#) chip_no
- u32 ccm_quotient_max

6.45.1 Detailed Description

Channel configuration structure.

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

The interpretation and validity of the remaining fields depend on the `usage` parameter.

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

Definition at line 1595 of file vtss_fdma_api.h.

6.45.2 Field Documentation

6.45.2.1 usage

```
vtss_fdma_ch_usage_t vtss_fdma_ch_cfg_t::usage
```

XTR/INJ/AFI:

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

Definition at line 1603 of file vtss_fdma_api.h.

6.45.2.2 xtr_grp

```
vtss_packet_rx_grp_t vtss_fdma_ch_cfg_t::xtr_grp
```

XTR:

The extraction group that this channel serves. Need only be set if cfg->chip_no == 1. Valid values are [0; VTSS_PACKET_RX_GRP_CNT[

INJ/AFI:

Unused.

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

Definition at line 1623 of file vtss_fdma_api.h.

6.45.2.3 inj_grp_mask

`u32 vtss_fdma_ch_cfg_t::inj_grp_mask`

XTR:

Unused.

INJ:

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

At least one bit must be set in the mask. The most significant bit that can be set is `VTSS_PACKET_TX_GRP_CNT - 1`.

AFI:

A mask containing the injection group that the AFI channel serves. This is a one-hot mask, that is, exactly one bit must be set, and the most significant bit that can be set is `VTSS_PACKET_TX_GRP_CNT - 1`.

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

Definition at line 1653 of file `vtss_fdma_api.h`.

6.45.2.4 list

`vtss_fdma_list_t* vtss_fdma_ch_cfg_t::list`

XTR:

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

INJ:

Unused.

AFI:

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

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

6.45.2.5 xtr_cb

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

XTR:

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

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

INJ/AFI:

Unused.

Definition at line 1709 of file vtss_fdma_api.h.

6.45.2.6 prio

```
int vtss_fdma_ch_cfg_t::prio
```

XTR/INJ/AFI:

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

Definition at line 1724 of file vtss_fdma_api.h.

6.45.2.7 chip_no

```
vtss_chip_no_t vtss_fdma_ch_cfg_t::chip_no
```

XTR

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

INJ/AFI:

Unused.

Definition at line 1745 of file vtss_fdma_api.h.

6.45.2.8 ccm_quotient_max

```
u32 vtss_fdma_ch_cfg_t::ccm_quotient_max
```

AFI:

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

Valid values are in the range [1; VTSS_FDMA_CCM_QUOTIENT_MAX].

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

Definition at line 1775 of file vtss_fdma_api.h.

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

- vtss_api/include/vtss_fdma_api.h

6.46 vtss_fdma_throttle_cfg_t Struct Reference

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

Data Fields

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

6.46.1 Detailed Description

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

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

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

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

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

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

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

There is no need to call [vtss_fdma_throttle_tick\(\)](#) unless throttling is enabled for at least one extraction queue.

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

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

Definition at line 2204 of file vtss_fdma_api.h.

6.46.2 Field Documentation

6.46.2.1 frm_limit_per_tick

`u32 vtss_fdma_throttle_cfg_t::frm_limit_per_tick [VTSS_PACKET_RX_QUEUE_CNT]`

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

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

Definition at line 2224 of file vtss_fdma_api.h.

6.46.2.2 byte_limit_per_tick

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

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

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

Definition at line 2235 of file vtss_fdma_api.h.

6.46.2.3 suspend_tick_cnt

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

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

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

Definition at line 2247 of file vtss_fdma_api.h.

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

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

6.47 vtss_fdma_tx_info_t Struct Reference

FDMA Injection Properties.

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

Data Fields

- void * [pre_cb_ctxt1](#)
- void * [pre_cb_ctxt2](#)
- void(* [pre_cb](#))(void *ctxt1, void *ctxt2, [vtss_fdma_list_t](#) *list)
- u32 [afi_fps](#)
- [vtss_fdma_afi_type_t](#) [afi_type](#)
- BOOL [afi_enable_counting](#)
- BOOL [afi_enable_sequence_numbering](#)
- u16 [afi_sequence_number_offset](#)

6.47.1 Detailed Description

FDMA Injection Properties.

Definition at line 742 of file vtss_fdma_api.h.

6.47.2 Field Documentation

6.47.2.1 pre_cb_ctxt1

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

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

Definition at line 746 of file vtss_fdma_api.h.

6.47.2.2 pre_cb_ctxt2

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

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

Definition at line 751 of file vtss_fdma_api.h.

6.47.2.3 pre_cb

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

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

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

Parameters:

- ctxt1: The value of [@pre_cb_ctxt1](#).
- ctxt2: The value of [@pre_cb_ctxt2](#).
- list: Pointer to the SOF item. Validity: Luton26: Y Jaguar1: Y Serval : Y Jaguar2: Y Serval2: Y ServalT: Y

Definition at line 779 of file vtss_fdma_api.h.

6.47.2.4 afi_fps

`u32 vtss_fdma_tx_info_t::afi_fps`

Number of times this frame will be injected per second.

There is a number of restrictions on AFI frames:

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

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

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

Definition at line 817 of file `vtss_fdma_api.h`.

6.47.2.5 afi_type

`vtss_fdma_afi_type_t vtss_fdma_tx_info_t::afi_type`

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

Currently any given platform only supports one or the other, so there's no reason to set it to anything but VTSS_←FDMA_AFI_TYPE_AUTO.

Definition at line 829 of file `vtss_fdma_api.h`.

6.47.2.6 afi_enable_counting

`BOOL vtss_fdma_tx_info_t::afi_enable_counting`

Indicates whether this frame is subject to counting.

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

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

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

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

Definition at line 856 of file `vtss_fdma_api.h`.

6.47.2.7 afi_enable_sequence_numbering

`BOOL vtss_fdma_tx_info_t::afi_enable_sequence_numbering`

Indicates whether this frame is subject to sequence number updating.

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

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

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

SO BE CAREFUL WHEN YOU DESIGN THE APPLICATION!

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

Definition at line 896 of file `vtss_fdma_api.h`.

6.47.2.8 afi_sequence_number_offset

`u16 vtss_fdma_tx_info_t::afi_sequence_number_offset`

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

It is only used when `afi_enable_sequence_numbering` is TRUE.

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

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

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

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

Definition at line 921 of file `vtss_fdma_api.h`.

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

- `vtss_api/include/vtss_fdma_api.h`

6.48 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`

6.48.1 Detailed Description

Initialization configuration.

Definition at line 515 of file vtss_init_api.h.

6.48.2 Field Documentation

6.48.2.1 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.

6.48.2.2 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.

6.48.2.3 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.

6.48.2.4 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.

6.48.2.5 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.

6.48.2.6 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.

6.48.2.7 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.

6.48.2.8 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.

6.48.2.9 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.

6.48.2.10 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.

6.48.2.11 warm_start_enable

```
BOOL vtss_init_conf_t::warm_start_enable
```

Allow warm start

Definition at line 535 of file vtss_init_api.h.

6.48.2.12 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`.

6.48.2.13 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`.

6.48.2.14 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`.

6.48.2.15 pi

`vtss_pi_conf_t` `vtss_init_conf_t::pi`

Parallel Interface configuration

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

6.48.2.16 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`

6.49 vtss_inst_create_t Struct Reference

Create structure.

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

Data Fields

- [vtss_target_type_t target](#)

6.49.1 Detailed Description

Create structure.

Definition at line 78 of file vtss_init_api.h.

6.49.2 Field Documentation

6.49.2.1 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](#)

6.50 vtss_intr_t Struct Reference

Interrupt source structure.

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

Data Fields

- [BOOL link_change \[VTSS_PORT_ARRAY_SIZE\]](#)

6.50.1 Detailed Description

Interrupt source structure.

Definition at line 913 of file vtss_misc_api.h.

6.50.2 Field Documentation

6.50.2.1 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](#)

6.51 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](#)

6.51.1 Detailed Description

Either an IPv4 or IPv6 address.

Definition at line 813 of file types.h.

6.51.2 Field Documentation

6.51.2.1 type

`vtss_ip_type_t vtss_ip_addr_t::type`

Union type

Definition at line 814 of file types.h.

6.51.2.2 ipv4

`vtss_ipv4_t vtss_ip_addr_t::ipv4`

IPv4 address

Definition at line 816 of file types.h.

6.51.2.3 ipv6

`vtss_ipv6_t vtss_ip_addr_t::ipv6`

IPv6 address

Definition at line 817 of file types.h.

6.51.2.4 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`

6.52 vtss_ip_network_t Struct Reference

IPv6 network.

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

Data Fields

- `vtss_ip_addr_t address`
- `vtss_prefix_size_t prefix_size`

6.52.1 Detailed Description

IPv6 network.

Definition at line 836 of file types.h.

6.52.2 Field Documentation

6.52.2.1 address

`vtss_ip_addr_t vtss_ip_network_t::address`

Network address

Definition at line 838 of file types.h.

6.52.2.2 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`

6.53 vtss_ipv4_network_t Struct Reference

IPv4 network.

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

Data Fields

- `vtss_ip4_t address`
- `vtss_prefix_size_t prefix_size`

6.53.1 Detailed Description

IPv4 network.

Definition at line 822 of file types.h.

6.53.2 Field Documentation

6.53.2.1 address

`vtss_ipv4_t vtss_ipv4_network_t::address`

Network address

Definition at line 824 of file types.h.

6.53.2.2 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`

6.54 vtss_ipv4_uc_t Struct Reference

IPv4 unicast routing entry.

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

Data Fields

- `vtss_ipv4_network_t network`
- `vtss_ipv4_t destination`

6.54.1 Detailed Description

IPv4 unicast routing entry.

Definition at line 852 of file types.h.

6.54.2 Field Documentation

6.54.2.1 network

`vtss_ipv4_network_t vtss_ipv4_uc_t::network`

Network to route

Definition at line 854 of file types.h.

6.54.2.2 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`

6.55 `vtss_ipv6_network_t` Struct Reference

IPv6 network.

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

Data Fields

- `vtss_ipv6_t address`
- `vtss_prefix_size_t prefix_size`

6.55.1 Detailed Description

IPv6 network.

Definition at line 829 of file types.h.

6.55.2 Field Documentation

6.55.2.1 address

`vtss_ipv6_t vtss_ipv6_network_t::address`

Network address

Definition at line 831 of file types.h.

6.55.2.2 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`

6.56 vtss_ipv6_t Struct Reference

IPv6 address/mask.

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

Data Fields

- `u8 addr [16]`

6.56.1 Detailed Description

IPv6 address/mask.

Definition at line 797 of file types.h.

6.56.2 Field Documentation

6.56.2.1 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](#)

6.57 vtss_ipv6_uc_t Struct Reference

IPv6 routing entry.

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

Data Fields

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

6.57.1 Detailed Description

IPv6 routing entry.

Definition at line 860 of file types.h.

6.57.2 Field Documentation

6.57.2.1 network

`vtss_ipv6_network_t vtss_ipv6_uc_t::network`

Network to route

Definition at line 862 of file types.h.

6.57.2.2 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`

6.58 vtss_irq_conf_t Struct Reference

Interrupt configuration options.

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

Data Fields

- `BOOL external`
- `u8 destination`

6.58.1 Detailed Description

Interrupt configuration options.

Definition at line 972 of file vtss_misc_api.h.

6.58.2 Field Documentation

6.58.2.1 external

`BOOL vtss_irq_conf_t::external`

Redirect to external IRQ

Definition at line 973 of file vtss_misc_api.h.

6.58.2.2 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`

6.59 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`

6.59.1 Detailed Description

Interrupt status structure.

Definition at line 980 of file vtss_misc_api.h.

6.59.2 Field Documentation

6.59.2.1 active

`u32 vtss_irq_status_t::active`

Bitmap for pending IRQs (VTSS_IRQ_xxx)

Definition at line 981 of file vtss_misc_api.h.

6.59.2.2 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.

6.59.2.3 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.

6.59.2.4 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

6.60 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

6.60.1 Detailed Description

Routing interface statics counter.

Definition at line 886 of file types.h.

6.60.2 Field Documentation

6.60.2.1 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.

6.60.2.2 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.

6.60.2.3 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.

6.60.2.4 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.

6.60.2.5 ipv4uc_transmitted_octets

`u64 vtss_l3_counters_t::ipv4uc_transmitted_octets`

IPv4UC octets transmitted

Definition at line 892 of file types.h.

6.60.2.6 ipv4uc_transmitted_frames

`u64 vtss_l3_counters_t::ipv4uc_transmitted_frames`

IPv4UC frames transmitted

Definition at line 893 of file types.h.

6.60.2.7 ipv6uc_transmitted_octets

`u64 vtss_l3_counters_t::ipv6uc_transmitted_octets`

IPv6UC octets transmitted

Definition at line 894 of file types.h.

6.60.2.8 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`

6.61 vtss_lcpll_status_t Struct Reference

Structure for Get PHY LC-PLL status.

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

Data Fields

- `u8 lock_status`
- `u8 cal_done`
- `u8 cal_error`
- `u8 fsm_lock`
- `u8 fsm_stat`
- `u8 gain_stat`

6.61.1 Detailed Description

Structure for Get PHY LC-PLL status.

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

6.61.2 Field Documentation

6.61.2.1 lock_status

`u8 vtss_lcppll_status_t::lock_status`

PLL lock status

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

6.61.2.2 cal_done

`u8 vtss_lcppll_status_t::cal_done`

Calibration status

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

6.61.2.3 cal_error

`u8 vtss_lcppll_status_t::cal_error`

Calibration Error indication

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

6.61.2.4 fsm_lock

`u8 vtss_lcp11_status_t::fsm_lock`

FSM lock status

Definition at line 1781 of file vtss_phy_api.h.

6.61.2.5 fsm_stat

`u8 vtss_lcp11_status_t::fsm_stat`

FSM internal status

Definition at line 1782 of file vtss_phy_api.h.

6.61.2.6 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](#)

6.62 vtss_learn_mode_t Struct Reference

Learning mode.

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

Data Fields

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

6.62.1 Detailed Description

Learning mode.

Definition at line 379 of file vtss_l2_api.h.

6.62.2 Field Documentation

6.62.2.1 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`.

6.62.2.2 cpu

`BOOL vtss_learn_mode_t::cpu`

Learn frames copied to CPU (default disabled)

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

6.62.2.3 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`

6.63 vtss_mac_t Struct Reference

MAC Address.

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

Data Fields

- `u8 addr [6]`

6.63.1 Detailed Description

MAC Address.

Definition at line 643 of file types.h.

6.63.2 Field Documentation

6.63.2.1 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](#)

6.64 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](#)

6.64.1 Detailed Description

MAC address entry.

Definition at line 119 of file vtss_l2_api.h.

6.64.2 Field Documentation

6.64.2.1 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.

6.64.2.2 destination

`BOOL vtss_mac_table_entry_t::destination[VTSS_PORT_ARRAY_SIZE]`

Dest. ports

Definition at line 122 of file vtss_l2_api.h.

6.64.2.3 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.

6.64.2.4 locked

`BOOL vtss_mac_table_entry_t::locked`

Locked/static flag

Definition at line 124 of file vtss_l2_api.h.

6.64.2.5 aged

`BOOL vtss_mac_table_entry_t::aged`

Age flag

Definition at line 125 of file vtss_l2_api.h.

6.64.2.6 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`

6.65 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`

6.65.1 Detailed Description

MAC address table status.

Definition at line 358 of file vtss_l2_api.h.

6.65.2 Field Documentation

6.65.2.1 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.

6.65.2.2 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`.

6.65.2.3 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`.

6.65.2.4 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`

6.66 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`

6.66.1 Detailed Description

MCE action.

Definition at line 759 of file vtss_evc_api.h.

6.66.2 Field Documentation

6.66.2.1 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.

6.66.2.2 prio_enable

`BOOL vtss_mce_action_t::prio_enable`

Enable priority control

Definition at line 777 of file vtss_evc_api.h.

6.66.2.3 prio

`vtss_prio_t vtss_mce_action_t::prio`

Selected priority

Definition at line 778 of file vtss_evc_api.h.

6.66.2.4 vid

`vtss_vid_t vtss_mce_action_t::vid`

Replace VID

Definition at line 779 of file vtss_evc_api.h.

6.66.2.5 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](#)

6.67 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](#)

6.67.1 Detailed Description

MCE key.

Definition at line 682 of file vtss_evc_api.h.

6.67.2 Field Documentation

6.67.2.1 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.

6.67.2.2 vid

`vtss_vcap_vid_t` `vtss_mce_key_t::vid`

Classified VID

Definition at line 697 of file vtss_evc_api.h.

6.67.2.3 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`

6.68 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`

6.68.1 Detailed Description

MEP Control Entry.

Definition at line 784 of file vtss_evc_api.h.

6.68.2 Field Documentation

6.68.2.1 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`.

6.68.2.2 id

`vtss_mce_id_t vtss_mce_t::id`

Entry ID

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

6.68.2.3 key

`vtss_mce_key_t vtss_mce_t::key`

MCE key

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

6.68.2.4 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`

6.69 vtss_mirror_conf_t Struct Reference

Mirror configuration.

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

Data Fields

- `vtss_port_no_t port_no`
- `BOOL fwd_enable`

6.69.1 Detailed Description

Mirror configuration.

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

6.69.2 Field Documentation

6.69.2.1 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`.

6.69.2.2 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`

6.70 vtss_mtimer_t Struct Reference

Timer structure.

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

Data Fields

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

6.70.1 Detailed Description

Timer structure.

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

6.70.2 Field Documentation

6.70.2.1 timeout

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

Timeout

Definition at line 89 of file `vtss_os_linux.h`.

6.70.2.2 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`

6.71 vtss_npi_conf_t Struct Reference

NPI configuration.

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

Data Fields

- `BOOL enable`
- `vtss_port_no_t port_no`

6.71.1 Detailed Description

NPI configuration.

Definition at line 48 of file vtss_packet_api.h.

6.71.2 Field Documentation

6.71.2.1 enable

`BOOL vtss_npi_conf_t::enable`

Enable NPI port

Definition at line 49 of file vtss_packet_api.h.

6.71.2.2 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

6.72 vtss_os_timestamp_t Struct Reference

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

Data Fields

- unsigned int `hw_cnt`

6.72.1 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.

6.72.2 Field Documentation

6.72.2.1 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](#)

6.73 vtss_packet_dma_conf_t Struct Reference

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

Data Fields

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

6.73.1 Detailed Description

The packet DMA configuration.

Definition at line 2124 of file vtss_packet_api.h.

6.73.2 Field Documentation

6.73.2.1 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](#)

6.74 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`

6.74.1 Detailed Description

Information about frame.

Definition at line 346 of file vtss_packet_api.h.

6.74.2 Field Documentation

6.74.2.1 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.

6.74.2.2 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.

6.74.2.3 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.

6.74.2.4 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.

6.74.2.5 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

6.75 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`

6.75.1 Detailed Description

Packet information for each port.

Definition at line 410 of file vtss_packet_api.h.

6.75.2 Field Documentation

6.75.2.1 filter

`vtss_packet_filter_t vtss_packet_port_filter_t::filter`

Packet filtering

Definition at line 411 of file vtss_packet_api.h.

6.75.2.2 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](#)

6.76 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`

6.76.1 Detailed Description

Port info structure.

Definition at line 390 of file vtss_packet_api.h.

6.76.2 Field Documentation

6.76.2.1 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.

6.76.2.2 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.

6.76.2.3 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.

6.76.2.4 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](#)

6.77 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]`

6.77.1 Detailed Description

CPU Rx configuration.

Definition at line 121 of file vtss_packet_api.h.

6.77.2 Field Documentation

6.77.2.1 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.

6.77.2.2 reg

```
vtss_packet_rx_reg_t vtss_packet_rx_conf_t::reg
```

Packet registration

Definition at line 123 of file vtss_packet_api.h.

6.77.2.3 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`.

6.77.2.4 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`

6.78 `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`

6.78.1 Detailed Description

System frame header describing received frame.

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

6.78.2 Field Documentation

6.78.2.1 length

`u32 vtss_packet_rx_header_t::length`

Frame length excluding CRC

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

6.78.2.2 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`.

6.78.2.3 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`.

6.78.2.4 learn

`BOOL vtss_packet_rx_header_t::learn`

TRUE if learn frame

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

6.78.2.5 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`.

6.78.2.6 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`

6.79 `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`

6.79.1 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`.

6.79.2 Field Documentation

6.79.2.1 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](#).

6.79.2.2 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](#).

6.79.2.3 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.

6.79.2.4 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.

6.79.2.5 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`.

6.79.2.6 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`.

6.79.2.7 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`.

6.79.2.8 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`.

6.79.2.9 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.

6.79.2.10 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.

6.79.2.11 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.

6.79.2.12 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`.

6.79.2.13 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`.

6.79.2.14 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`.

6.79.2.15 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.

6.79.2.16 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.

6.79.2.17 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.

6.79.2.18 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←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.

6.79.2.19 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.

6.79.2.20 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.

6.79.2.21 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`

6.80 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`

6.80.1 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.

6.80.2 Field Documentation

6.80.2.1 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.

6.80.2.2 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.

6.80.2.3 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.

6.80.2.4 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.

6.80.2.5 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`.

6.80.2.6 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`.

6.80.2.7 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`

6.81 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]

6.81.1 Detailed Description

Packet registration per port.

Definition at line 764 of file types.h.

6.81.2 Field Documentation

6.81.2.1 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.

6.81.2.2 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`

6.82 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`

6.82.1 Detailed Description

CPU Rx queue configuration.

Definition at line 85 of file vtss_packet_api.h.

6.82.2 Field Documentation

6.82.2.1 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.

6.82.2.2 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`

6.83 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`

6.83.1 Detailed Description

CPU Rx queue map.

Definition at line 103 of file vtss_packet_api.h.

6.83.2 Field Documentation

6.83.2.1 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.

6.83.2.2 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.

6.83.2.3 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.

6.83.2.4 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.

6.83.2.5 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.

6.83.2.6 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`.

6.83.2.7 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`.

6.83.2.8 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`.

6.83.2.9 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`

6.84 **vtss_packet_rx_queue_npi_conf_t** Struct Reference

CPU Rx queue NPI configuration.

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

Data Fields

- `BOOL enable`

6.84.1 Detailed Description

CPU Rx queue NPI configuration.

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

6.84.2 Field Documentation

6.84.2.1 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`

6.85 `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`

6.85.1 Detailed Description

CPU Rx packet registration.

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

6.85.2 Field Documentation

6.85.2.1 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.

6.85.2.2 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.

6.85.2.3 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.

6.85.2.4 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.

6.85.2.5 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`

6.86 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]`

6.86.1 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`.

6.86.2 Field Documentation

6.86.2.1 length

`u32 vtss_packet_tx_ifh_t::length`

Length of compiled IFH (in bytes)

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

6.86.2.2 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

6.87 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](#)

6.87.1 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.

6.87.2 Field Documentation

6.87.2.1 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`.

6.87.2.2 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.

6.87.2.3 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.

6.87.2.4 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.

6.87.2.5 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.

6.87.2.6 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.

6.87.2.7 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.

6.87.2.8 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.

6.87.2.9 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.

6.87.2.10 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.

6.87.2.11 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.

6.87.2.12 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`.

6.87.2.13 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`.

6.87.2.14 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.

6.87.2.15 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.

6.87.2.16 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](#)

6.88 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](#)

6.88.1 Detailed Description

PHY auto negotiation advertisement.

Definition at line 309 of file vtss_phy_api.h.

6.88.2 Field Documentation

6.88.2.1 speed_10m_hdx

`BOOL vtss_phy_aneg_t::speed_10m_hdx`

10Mbps, half duplex

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

6.88.2.2 speed_10m_fdx

`BOOL vtss_phy_aneg_t::speed_10m_fdx`

10Mbps, full duplex

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

6.88.2.3 speed_100m_hdx

`BOOL vtss_phy_aneg_t::speed_100m_hdx`

100Mbps, half duplex

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

6.88.2.4 speed_100m_fdx

`BOOL vtss_phy_aneg_t::speed_100m_fdx`

100Mbps, full duplex

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

6.88.2.5 speed_1g_fdx

`BOOL vtss_phy_aneg_t::speed_1g_fdx`

1000Mbps, full duplex

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

6.88.2.6 speed_1g_hdx

`BOOL vtss_phy_aneg_t::speed_1g_hdx`

1000Mbps, full duplex

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

6.88.2.7 symmetric_pause

`BOOL vtss_phy_aneg_t::symmetric_pause`

Symmetric pause

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

6.88.2.8 asymmetric_pause

`BOOL vtss_phy_aneg_t::asymmetric_pause`

Asymmetric pause

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

6.88.2.9 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`

6.89 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`

6.89.1 Detailed Description

PHY clock configuration.

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

6.89.2 Field Documentation

6.89.2.1 `src`

`vtss_phy_clk_source_t` `vtss_phy_clock_conf_t::src`

Clock source

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

6.89.2.2 `freq`

`vtss_phy_freq_t` `vtss_phy_clock_conf_t::freq`

Clock frequency

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

6.89.2.3 `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`

6.90 vtss_phy_conf_1g_t Struct Reference

PHY 1G configuration.

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

Data Fields

- struct {
 BOOL cfg
 BOOL val
} master

6.90.1 Detailed Description

PHY 1G configuration.

Definition at line 504 of file vtss_phy_api.h.

6.90.2 Field Documentation

6.90.2.1 cfg

```
BOOL vtss_phy_conf_1g_t::cfg
```

Manual Master/Slave Config. 1=enabled

Definition at line 506 of file vtss_phy_api.h.

6.90.2.2 val

```
BOOL vtss_phy_conf_1g_t::val
```

Master/Slave Config value, 1=Master

Definition at line 507 of file vtss_phy_api.h.

6.90.2.3 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](#)

6.91 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](#)

6.91.1 Detailed Description

PHY configuration.

Definition at line 395 of file [vtss_phy_api.h](#).

6.91.2 Field Documentation

6.91.2.1 mode

```
vtss_phy_mode_t vtss_phy_conf_t::mode
```

PHY mode

Definition at line 396 of file [vtss_phy_api.h](#).

6.91.2.2 forced

`vtss_phy_forced_t` `vtss_phy_conf_t::forced`

Forced mode configuration

Definition at line 397 of file vtss_phy_api.h.

6.91.2.3 aneg

`vtss_phy_aneg_t` `vtss_phy_conf_t::aneg`

Auto-negotiation mode configuration

Definition at line 398 of file vtss_phy_api.h.

6.91.2.4 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.

6.91.2.5 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.

6.91.2.6 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.

6.91.2.7 unidir

`vtss_phy_unidirectional_t` `vtss_phy_conf_t::unidir`

Unidirectional Configuration

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

6.91.2.8 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`.

6.91.2.9 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`.

6.91.2.10 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`.

6.91.2.11 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`

6.92 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`

6.92.1 Detailed Description

EEE configuration.

Definition at line 987 of file vtss_phy_api.h.

6.92.2 Field Documentation

6.92.2.1 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.

6.92.2.2 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](#)

6.93 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`

6.93.1 Detailed Description

enhanced LED control

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

6.93.2 Field Documentation

6.93.2.1 `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`.

6.93.2.2 `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`.

6.93.2.3 `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`.

6.93.2.4 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](#)

6.94 vtss_phy_forced_t Struct Reference

PHY forced mode configuration.

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

Data Fields

- [vtss_port_speed_t speed](#)
- [BOOL fdx](#)

6.94.1 Detailed Description

PHY forced mode configuration.

Definition at line 303 of file vtss_phy_api.h.

6.94.2 Field Documentation

6.94.2.1 speed

```
vtss_port_speed_t vtss_phy_forced_t::speed
```

Speed

Definition at line 304 of file vtss_phy_api.h.

6.94.2.2 `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`

6.95 `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`

6.95.1 Detailed Description

LED model selection.

Definition at line 136 of file `vtss_phy_api.h`.

6.95.2 Field Documentation

6.95.2.1 `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`.

6.95.2.2 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`

6.96 `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`

6.96.1 Detailed Description

1G Phy loopbacks

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

6.96.2 Field Documentation

6.96.2.1 `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`.

6.96.2.2 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.

6.96.2.3 connector_enable

`BOOL vtss_phy_loopback_t::connector_enable`

Enable/Disable loopback connector loopback

Definition at line 1387 of file vtss_phy_api.h.

6.96.2.4 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.

6.96.2.5 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.

6.96.2.6 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.

6.96.2.7 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`.

6.96.2.8 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`.

6.96.2.9 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`

6.97 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`

6.97.1 Detailed Description

PHY MAC SerDes PCS Control, Reg16E3.

Definition at line 351 of file vtss_phy_api.h.

6.97.2 Field Documentation

6.97.2.1 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.

6.97.2.2 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.

6.97.2.3 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.

6.97.2.4 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.

6.97.2.5 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`.

6.97.2.6 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`.

6.97.2.7 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`.

6.97.2.8 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`.

6.97.2.9 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`.

6.97.2.10 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`.

6.97.2.11 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`.

6.97.2.12 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`

6.98 **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`

6.98.1 Detailed Description

PHY MEDIA SerDes PCS Control, Reg23E3.

Definition at line 375 of file vtss_phy_api.h.

6.98.2 Field Documentation

6.98.2.1 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.

6.98.2.2 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.

6.98.2.3 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.

6.98.2.4 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.

6.98.2.5 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`.

6.98.2.6 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`.

6.98.2.7 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`.

6.98.2.8 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`.

6.98.2.9 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`

6.99 vtss_phy_power_conf_t Struct Reference

PHY power configuration.

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

Data Fields

- [vtss_phy_power_mode_t mode](#)

6.99.1 Detailed Description

PHY power configuration.

Definition at line 562 of file vtss_phy_api.h.

6.99.2 Field Documentation

6.99.2.1 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](#)

6.100 vtss_phy_power_status_t Struct Reference

PHY power status.

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

Data Fields

- [u32 level](#)

6.100.1 Detailed Description

PHY power status.

Definition at line 597 of file vtss_phy_api.h.

6.100.2 Field Documentation

6.100.2.1 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

6.101 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`

6.101.1 Detailed Description

PHY reset structure.

Definition at line 218 of file vtss_phy_api.h.

6.101.2 Field Documentation

6.101.2.1 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.

6.101.2.2 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.

6.101.2.3 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.

6.101.2.4 tbi

`vtss_phy_tbi_conf_t` `vtss_phy_reset_conf_t::tbi`

TBI setup

Definition at line 222 of file vtss_phy_api.h.

6.101.2.5 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.

6.101.2.6 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.

6.101.2.7 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](#)

6.102 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`

6.102.1 Detailed Description

PHY RGMII configuration.

Definition at line 194 of file vtss_phy_api.h.

6.102.2 Field Documentation

6.102.2.1 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.

6.102.2.2 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](#)

6.103 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`

6.103.1 Detailed Description

Phy statistic information.

Definition at line 1348 of file vtss_phy_api.h.

6.103.2 Field Documentation

6.103.2.1 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.

6.103.2.2 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.

6.103.2.3 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.

6.103.2.4 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.

6.103.2.5 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.

6.103.2.6 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.

6.103.2.7 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](#)

6.104 vtss_phy_status_1g_t Struct Reference

PHY 1G status.

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

Data Fields

- [BOOL master_cfg_fault](#)
- [BOOL master](#)

6.104.1 Detailed Description

PHY 1G status.

Definition at line 538 of file vtss_phy_api.h.

6.104.2 Field Documentation

6.104.2.1 master_cfg_fault

`BOOL vtss_phy_status_lg_t::master_cfg_fault`

Master/Slave Configuration fault

Definition at line 539 of file vtss_phy_api.h.

6.104.2.2 master

`BOOL vtss_phy_status_lg_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`

6.105 vtss_phy_tbi_conf_t Struct Reference

PHY TBI configuration.

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

Data Fields

- `BOOL aneg_enable`

6.105.1 Detailed Description

PHY TBI configuration.

Definition at line 200 of file vtss_phy_api.h.

6.105.2 Field Documentation

6.105.2.1 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`

6.106 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`

6.106.1 Detailed Description

Phy type information.

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

6.106.2 Field Documentation

6.106.2.1 part_number

`u16 vtss_phy_type_t::part_number`

Part number

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

6.106.2.2 revision

`u16 vtss_phy_type_t::revision`

Chip revision

Definition at line 146 of file vtss_phy_api.h.

6.106.2.3 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.

6.106.2.4 channel_id

`u16 vtss_phy_type_t::channel_id`

Channel id

Definition at line 148 of file vtss_phy_api.h.

6.106.2.5 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.

6.106.2.6 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](#)

6.107 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]`

6.107.1 Detailed Description

VeriPHY result.

Definition at line 958 of file vtss_phy_api.h.

6.107.2 Field Documentation

6.107.2.1 link

```
BOOL vtss_phy_veriphy_result_t::link
```

Link status

Definition at line 959 of file vtss_phy_api.h.

6.107.2.2 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.

6.107.2.3 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`

6.108 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`

6.108.1 Detailed Description

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

Definition at line 1680 of file vtss_phy_api.h.

6.108.2 Field Documentation

6.108.2.1 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.

6.108.2.2 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.

6.108.2.3 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.

6.108.2.4 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.

6.108.2.5 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](#)

6.109 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`

6.109.1 Detailed Description

PI configuration.

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

6.109.2 Field Documentation

6.109.2.1 width

`vtss_pi_width_t vtss_pi_conf_t::width`

Width

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

6.109.2.2 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`.

6.109.2.3 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`

6.110 vtss_policer_ext_t Struct Reference

Policer Extensions.

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

Data Fields

- `BOOL frame_rate`
- `BOOL flow_control`

6.110.1 Detailed Description

Policer Extensions.

Definition at line 198 of file vtss_qos_api.h.

6.110.2 Field Documentation

6.110.2.1 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.

6.110.2.2 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

6.111 vtss_policer_t Struct Reference

Policer.

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

Data Fields

- `vtss_burst_level_t` `level`
- `vtss_bitrate_t` `rate`

6.111.1 Detailed Description

Policer.

Definition at line 185 of file `vtss_qos_api.h`.

6.111.2 Field Documentation

6.111.2.1 `level`

`vtss_burst_level_t` `vtss_policer_t::level`

Burst level

Definition at line 187 of file `vtss_qos_api.h`.

6.111.2.2 `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`

6.112 `vtss_port_bridge_counters_t` Struct Reference

Port bridge counter structure (RFC 4188)

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

Data Fields

- `vtss_port_counter_t` `dot1dTpPortInDiscards`

6.112.1 Detailed Description

Port bridge counter structure (RFC 4188)

Definition at line 201 of file port.h.

6.112.2 Field Documentation

6.112.2.1 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](#)

6.113 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](#)

6.113.1 Detailed Description

Advertisement control data for Clause 37 aneg.

Definition at line 127 of file vtss_port_api.h.

6.113.2 Field Documentation

6.113.2.1 fdx

`BOOL vtss_port_clause_37_adv_t::fdx`

(FD)

Definition at line 129 of file vtss_port_api.h.

6.113.2.2 hdx

`BOOL vtss_port_clause_37_adv_t::hdx`

(HD)

Definition at line 130 of file vtss_port_api.h.

6.113.2.3 symmetric_pause

`BOOL vtss_port_clause_37_adv_t::symmetric_pause`

(PS1)

Definition at line 131 of file vtss_port_api.h.

6.113.2.4 asymmetric_pause

`BOOL vtss_port_clause_37_adv_t::asymmetric_pause`

(PS2)

Definition at line 132 of file vtss_port_api.h.

6.113.2.5 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.

6.113.2.6 acknowledge

`BOOL vtss_port_clause_37_adv_t::acknowledge`

(Ack)

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

6.113.2.7 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`

6.114 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`

6.114.1 Detailed Description

Auto-negotiation control parameter struct.

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

6.114.2 Field Documentation

6.114.2.1 enable

`BOOL vtss_port_clause_37_control_t::enable`

Enable of Autoneg

Definition at line 154 of file vtss_port_api.h.

6.114.2.2 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`

6.115 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`

6.115.1 Detailed Description

Port configuration structure.

Definition at line 268 of file vtss_port_api.h.

6.115.2 Field Documentation

6.115.2.1 if_type

`vtss_port_interface_t` `vtss_port_conf_t::if_type`

Interface type

Definition at line 270 of file vtss_port_api.h.

6.115.2.2 sd_enable

`BOOL` `vtss_port_conf_t::sd_enable`

Signal detect enable

Definition at line 271 of file vtss_port_api.h.

6.115.2.3 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.

6.115.2.4 sd_internal

`BOOL` `vtss_port_conf_t::sd_internal`

Internal signal detect selection

Definition at line 273 of file vtss_port_api.h.

6.115.2.5 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.

6.115.2.6 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.

6.115.2.7 speed

`vtss_port_speed_t` `vtss_port_conf_t::speed`

Port speed

Definition at line 276 of file vtss_port_api.h.

6.115.2.8 fdx

`BOOL` `vtss_port_conf_t::fdx`

Full duplex mode

Definition at line 277 of file vtss_port_api.h.

6.115.2.9 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.

6.115.2.10 max_frame_length

`u32 vtss_port_conf_t::max_frame_length`

Maximum frame length

Definition at line 279 of file vtss_port_api.h.

6.115.2.11 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.

6.115.2.12 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.

6.115.2.13 exc_col_cont

`BOOL vtss_port_conf_t::exc_col_cont`

Excessive collision continuation

Definition at line 282 of file vtss_port_api.h.

6.115.2.14 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.

6.115.2.15 `xau_tx_lane_flip`

`BOOL vtss_port_conf_t::xau_tx_lane_flip`

Xau Tx lane flip

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

6.115.2.16 `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`.

6.115.2.17 `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`

6.116 `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`

6.116.1 Detailed Description

Port counter structure.

Definition at line 220 of file port.h.

6.116.2 Field Documentation

6.116.2.1 rmon

```
vtss_port_rmon_counters_t vtss_port_counters_t::rmon
```

RMON counters

Definition at line 222 of file port.h.

6.116.2.2 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.

6.116.2.3 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.

6.116.2.4 bridge

```
vtss_port_bridge_counters_t vtss_port_counters_t::bridge
```

Bridge counters

Definition at line 227 of file port.h.

6.116.2.5 prop

`vtss_port_proprietary_counters_t vtss_port_counters_t::prop`

Proprietary counters

Definition at line 230 of file port.h.

6.116.2.6 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`

6.117 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`

6.117.1 Detailed Description

Ethernet-like Interface counter structure (RFC 3635)

Definition at line 163 of file port.h.

6.117.2 Field Documentation

6.117.2.1 dot3InPauseFrames

`vtss_port_counter_t vtss_port_ethernet_like_counters_t::dot3InPauseFrames`

Rx pause

Definition at line 171 of file port.h.

6.117.2.2 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`

6.118 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]`

6.118.1 Detailed Description

EVC counters.

Definition at line 186 of file port.h.

6.118.2 Field Documentation

6.118.2.1 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.

6.118.2.2 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.

6.118.2.3 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.

6.118.2.4 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.

6.118.2.5 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.

6.118.2.6 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.

6.118.2.7 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](#)

6.119 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 \[VTSS_PRIOS\]](#)

6.119.1 Detailed Description

Flow control setup.

Definition at line 189 of file vtss_port_api.h.

6.119.2 Field Documentation

6.119.2.1 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.

6.119.2.2 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.

6.119.2.3 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.

6.119.2.4 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](#)

6.120 **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`

6.120.1 Detailed Description

Inter frame gap structure.

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

6.120.2 Field Documentation

6.120.2.1 `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`.

6.120.2.2 `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`.

6.120.2.3 `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`

6.121 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](#)

6.121.1 Detailed Description

Interfaces Group counter structure (RFC 2863)

Definition at line 144 of file port.h.

6.121.2 Field Documentation

6.121.2.1 ifInOctets

```
vtss_port_counter_t vtss_port_if_group_counters_t::ifInOctets
```

Rx octets

Definition at line 145 of file port.h.

6.121.2.2 ifInUcastPkts

```
vtss_port_counter_t vtss_port_if_group_counters_t::ifInUcastPkts
```

Rx unicasts

Definition at line 146 of file port.h.

6.121.2.3 ifInMulticastPkts

`vtss_port_counter_t vtss_port_if_group_counters_t::ifInMulticastPkts`

Rx multicasts

Definition at line 147 of file port.h.

6.121.2.4 ifInBroadcastPkts

`vtss_port_counter_t vtss_port_if_group_counters_t::ifInBroadcastPkts`

Rx broadcasts

Definition at line 148 of file port.h.

6.121.2.5 ifInNUcastPkts

`vtss_port_counter_t vtss_port_if_group_counters_t::ifInNUcastPkts`

Rx non-unicasts

Definition at line 149 of file port.h.

6.121.2.6 ifInDiscards

`vtss_port_counter_t vtss_port_if_group_counters_t::ifInDiscards`

Rx discards

Definition at line 150 of file port.h.

6.121.2.7 ifInErrors

`vtss_port_counter_t vtss_port_if_group_counters_t::ifInErrors`

Rx errors

Definition at line 151 of file port.h.

6.121.2.8 ifOutOctets

```
vtss_port_counter_t vtss_port_if_group_counters_t::ifOutOctets
```

Tx octets

Definition at line 153 of file port.h.

6.121.2.9 ifOutUcastPkts

```
vtss_port_counter_t vtss_port_if_group_counters_t::ifOutUcastPkts
```

Tx unicasts

Definition at line 154 of file port.h.

6.121.2.10 ifOutMulticastPkts

```
vtss_port_counter_t vtss_port_if_group_counters_t::ifOutMulticastPkts
```

Tx multicasts

Definition at line 155 of file port.h.

6.121.2.11 ifOutBroadcastPkts

```
vtss_port_counter_t vtss_port_if_group_counters_t::ifOutBroadcastPkts
```

Tx broadcasts

Definition at line 156 of file port.h.

6.121.2.12 ifOutNUcastPkts

```
vtss_port_counter_t vtss_port_if_group_counters_t::ifOutNUcastPkts
```

Tx non-unicasts

Definition at line 157 of file port.h.

6.121.2.13 ifOutDiscards

```
vtss_port_counter_t vtss_port_if_group_counters_t::ifOutDiscards
```

Tx discards

Definition at line 158 of file port.h.

6.121.2.14 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](#)

6.122 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](#)

6.122.1 Detailed Description

Port map structure.

Definition at line 72 of file vtss_port_api.h.

6.122.2 Field Documentation

6.122.2.1 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.

6.122.2.2 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.

6.122.2.3 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.

6.122.2.4 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.

6.122.2.5 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](#)

6.123 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]`

6.123.1 Detailed Description

Port proprietary counter structure.

Definition at line 207 of file port.h.

6.123.2 Field Documentation

6.123.2.1 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.

6.123.2.2 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`

6.124 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`

6.124.1 Detailed Description

RMON counter structure (RFC 2819)

Definition at line 109 of file port.h.

6.124.2 Field Documentation

6.124.2.1 rx_etherStatsDropEvents

```
vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsDropEvents
```

Rx drop events

Definition at line 110 of file port.h.

6.124.2.2 rx_etherStatsOctets

`vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsOctets`

Rx octets

Definition at line 111 of file port.h.

6.124.2.3 rx_etherStatsPkts

`vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsPkts`

Rx packets

Definition at line 112 of file port.h.

6.124.2.4 rx_etherStatsBroadcastPkts

`vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsBroadcastPkts`

Rx broadcasts

Definition at line 113 of file port.h.

6.124.2.5 rx_etherStatsMulticastPkts

`vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsMulticastPkts`

Rx multicasts

Definition at line 114 of file port.h.

6.124.2.6 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.

6.124.2.7 rx_etherStatsUndersizePkts

```
vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsUndersizePkts
```

Rx undersize packets

Definition at line 116 of file port.h.

6.124.2.8 rx_etherStatsOversizePkts

```
vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsOversizePkts
```

Rx oversize packets

Definition at line 117 of file port.h.

6.124.2.9 rx_etherStatsFragments

```
vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsFragments
```

Rx fragments

Definition at line 118 of file port.h.

6.124.2.10 rx_etherStatsJabbers

```
vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsJabbers
```

Rx jabbers

Definition at line 119 of file port.h.

6.124.2.11 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.

6.124.2.12 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.

6.124.2.13 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.

6.124.2.14 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.

6.124.2.15 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.

6.124.2.16 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.

6.124.2.17 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.

6.124.2.18 tx_etherStatsDropEvents

```
vtss_port_counter_t vtss_port_rmon_counters_t::tx_etherStatsDropEvents
```

Tx drop events

Definition at line 128 of file port.h.

6.124.2.19 tx_etherStatsOctets

```
vtss_port_counter_t vtss_port_rmon_counters_t::tx_etherStatsOctets
```

Tx octets

Definition at line 129 of file port.h.

6.124.2.20 tx_etherStatsPkts

```
vtss_port_counter_t vtss_port_rmon_counters_t::tx_etherStatsPkts
```

Tx packets

Definition at line 130 of file port.h.

6.124.2.21 tx_etherStatsBroadcastPkts

```
vtss_port_counter_t vtss_port_rmon_counters_t::tx_etherStatsBroadcastPkts
```

Tx broadcasts

Definition at line 131 of file port.h.

6.124.2.22 tx_etherStatsMulticastPkts

```
vtss_port_counter_t vtss_port_rmon_counters_t::tx_etherStatsMulticastPkts
```

Tx multicasts

Definition at line 132 of file port.h.

6.124.2.23 tx_etherStatsCollisions

```
vtss_port_counter_t vtss_port_rmon_counters_t::tx_etherStatsCollisions
```

Tx collisions

Definition at line 133 of file port.h.

6.124.2.24 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.

6.124.2.25 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.

6.124.2.26 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.

6.124.2.27 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.

6.124.2.28 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.

6.124.2.29 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.

6.124.2.30 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](#)

6.125 vtss_port_serdes_conf_t Struct Reference

SFI Serdes configuration.

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

Data Fields

- `BOOL sfp_dac`

6.125.1 Detailed Description

SFI Serdes configuration.

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

6.125.2 Field Documentation

6.125.2.1 `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`

6.126 `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`

6.126.1 Detailed Description

Advertisement control data for SGMII aneg.

Definition at line 139 of file `vtss_port_api.h`.

6.126.2 Field Documentation

6.126.2.1 link

`BOOL vtss_port_sgmii_aneg_t::link`

LP link status

Definition at line 141 of file `vtss_port_api.h`.

6.126.2.2 fdx

`BOOL vtss_port_sgmii_aneg_t::fdx`

FD

Definition at line 142 of file `vtss_port_api.h`.

6.126.2.3 hdx

`BOOL vtss_port_sgmii_aneg_t::hdx`

HD

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

6.126.2.4 speed_10M

`BOOL vtss_port_sgmii_aneg_t::speed_10M`

speed 10 advertised

Definition at line 144 of file `vtss_port_api.h`.

6.126.2.5 speed_100M

`BOOL vtss_port_sgmii_aneg_t::speed_100M`

speed 100 advertised

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

6.126.2.6 speed_1G

`BOOL vtss_port_sgmii_aneg_t::speed_1G`

speed 1G advertised

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

6.126.2.7 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`

6.127 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`

6.127.1 Detailed Description

Port status parameter struct.

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

6.127.2 Field Documentation

6.127.2.1 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.

6.127.2.2 link

`BOOL vtss_port_status_t::link`

Link is up. Remaining fields only valid if TRUE

Definition at line 298 of file port.h.

6.127.2.3 speed

`vtss_port_speed_t vtss_port_status_t::speed`

Speed

Definition at line 299 of file port.h.

6.127.2.4 fdx

`BOOL vtss_port_status_t::fdx`

Full duplex

Definition at line 300 of file port.h.

6.127.2.5 remote_fault

`BOOL vtss_port_status_t::remote_fault`

Remote fault signalled

Definition at line 301 of file port.h.

6.127.2.6 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.

6.127.2.7 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.

6.127.2.8 aneg

`vtss_aneg_t vtss_port_status_t::aneg`

Auto negotiation result

Definition at line 307 of file port.h.

6.127.2.9 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.

6.127.2.10 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.

6.127.2.11 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`

6.128 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`

6.128.1 Detailed Description

QCE action.

Definition at line 566 of file vtss_qos_api.h.

6.128.2 Field Documentation

6.128.2.1 prio_enable

`BOOL vtss_qce_action_t::prio_enable`

Enable priority classification

Definition at line 568 of file vtss_qos_api.h.

6.128.2.2 prio

`vtss_prio_t vtss_qce_action_t::prio`

Priority value

Definition at line 569 of file vtss_qos_api.h.

6.128.2.3 dp_enable

`BOOL vtss_qce_action_t::dp_enable`

Enable DP classification

Definition at line 570 of file vtss_qos_api.h.

6.128.2.4 dp

`vtss_dp_level_t vtss_qce_action_t::dp`

DP value

Definition at line 571 of file vtss_qos_api.h.

6.128.2.5 dscp_enable

`BOOL vtss_qce_action_t::dscp_enable`

Enable DSCP classification

Definition at line 572 of file vtss_qos_api.h.

6.128.2.6 dscp

`vtss_dscp_t vtss_qce_action_t::dscp`

DSCP value

Definition at line 573 of file vtss_qos_api.h.

6.128.2.7 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.

6.128.2.8 pcp

`vtss_tagprio_t vtss_qce_action_t::pcp`

PCP value

Definition at line 576 of file vtss_qos_api.h.

6.128.2.9 dei

`vtss_dei_t vtss_qce_action_t::dei`

DEI value

Definition at line 577 of file vtss_qos_api.h.

6.128.2.10 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.

6.128.2.11 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](#)

6.129 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`

6.129.1 Detailed Description

Frame data for VTSS_QCE_TYPEETYPE.

Definition at line 494 of file vtss_qos_api.h.

6.129.2 Field Documentation

6.129.2.1 etype

```
vtss_vcap_u16_t vtss_qce_frame_etype_t::etype
```

Ethernet Type value

Definition at line 496 of file vtss_qos_api.h.

6.129.2.2 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

6.130 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

6.130.1 Detailed Description

Frame data for VTSS_QCE_TYPE_IPV4.

Definition at line 513 of file vtss_qos_api.h.

6.130.2 Field Documentation

6.130.2.1 fragment

`vtss_vcap_bit_t` vtss_qce_frame_ipv4_t::fragment

Fragment

Definition at line 515 of file vtss_qos_api.h.

6.130.2.2 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.

6.130.2.3 proto

`vtss_vcap_u8_t` vtss_qce_frame_ipv4_t::proto

Protocol

Definition at line 517 of file vtss_qos_api.h.

6.130.2.4 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.

6.130.2.5 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.

6.130.2.6 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](#)

6.131 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`

6.131.1 Detailed Description

Frame data for VTSS_QCE_TYPE_IPV6.

Definition at line 527 of file vtss_qos_api.h.

6.131.2 Field Documentation

6.131.2.1 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.

6.131.2.2 proto

`vtss_vcap_u8_t` vtss_qce_frame_ipv6_t::proto

Protocol

Definition at line 530 of file vtss_qos_api.h.

6.131.2.3 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.

6.131.2.4 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.

6.131.2.5 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`

6.132 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`

6.132.1 Detailed Description

Frame data for VTSS_QCE_TYPE LLC.

Definition at line 501 of file vtss_qos_api.h.

6.132.2 Field Documentation

6.132.2.1 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`

6.133 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

6.133.1 Detailed Description

Frame data for VTSS_QCE_TYPE_SNAP.

Definition at line 507 of file vtss_qos_api.h.

6.133.2 Field Documentation

6.133.2.1 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

6.134 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

6.134.1 Detailed Description

QCE key.

Definition at line 542 of file vtss_qos_api.h.

6.134.2 Field Documentation

6.134.2.1 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.

6.134.2.2 mac

```
vtss_qce_mac_t vtss_qce_key_t::mac
```

MAC

Definition at line 545 of file vtss_qos_api.h.

6.134.2.3 tag

```
vtss_qce_tag_t vtss_qce_key_t::tag
```

Tag

Definition at line 546 of file vtss_qos_api.h.

6.134.2.4 type

```
vtss_qce_type_t vtss_qce_key_t::type
```

Frame type

Definition at line 550 of file vtss_qos_api.h.

6.134.2.5 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.

6.134.2.6 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.

6.134.2.7 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.

6.134.2.8 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.

6.134.2.9 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.

6.134.2.10 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](#)

6.135 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](#)

6.135.1 Detailed Description

QCE MAC information.

Definition at line 471 of file [vtss_qos_api.h](#).

6.135.2 Field Documentation

6.135.2.1 dmac_mc

```
vtss\_vcap\_bit\_t vtss_qce_mac_t::dmac_mc
```

Multicast DMAC

Definition at line 473 of file [vtss_qos_api.h](#).

6.135.2.2 dmac_bc

`vtss_vcap_bit_t` `vtss_qce_mac_t::dmac_bc`

Broadcast DMAC

Definition at line 474 of file `vtss_qos_api.h`.

6.135.2.3 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`

6.136 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`

6.136.1 Detailed Description

QoS Control Entry.

Definition at line 588 of file `vtss_qos_api.h`.

6.136.2 Field Documentation

6.136.2.1 id

`vtss_qce_id_t` `vtss_qce_t::id`

Entry ID

Definition at line 590 of file `vtss_qos_api.h`.

6.136.2.2 key

`vtss_qce_key_t` `vtss_qce_t::key`

QCE key

Definition at line 591 of file `vtss_qos_api.h`.

6.136.2.3 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`

6.137 `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`

6.137.1 Detailed Description

QCE tag information.

Definition at line 482 of file vtss_qos_api.h.

6.137.2 Field Documentation

6.137.2.1 vid

`vtss_vcap_vr_t` `vtss_qce_tag_t::vid`

VLAN ID (12 bit)

Definition at line 484 of file vtss_qos_api.h.

6.137.2.2 pcp

`vtss_vcap_u8_t` `vtss_qce_tag_t::pcp`

PCP (3 bit)

Definition at line 485 of file vtss_qos_api.h.

6.137.2.3 dei

`vtss_vcap_bit_t` `vtss_qce_tag_t::dei`

DEI

Definition at line 486 of file vtss_qos_api.h.

6.137.2.4 tagged

`vtss_vcap_bit_t` `vtss_qce_tag_t::tagged`

Tagged/untagged frame

Definition at line 487 of file vtss_qos_api.h.

6.137.2.5 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](#)

6.138 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

6.138.1 Detailed Description

All parameters below are defined per chip.

Definition at line 91 of file vtss_qos_api.h.

6.138.2 Field Documentation

6.138.2.1 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.

6.138.2.2 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.

6.138.2.3 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.

6.138.2.4 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.

6.138.2.5 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.

6.138.2.6 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.

6.138.2.7 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.

6.138.2.8 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.

6.138.2.9 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.

6.138.2.10 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.

6.138.2.11 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.

6.138.2.12 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.

6.138.2.13 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.

6.138.2.14 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.

6.138.2.15 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.

6.138.2.16 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`.

6.138.2.17 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`.

6.138.2.18 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`.

6.138.2.19 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`

6.139 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`

6.139.1 Detailed Description

QoS setup per port.

Definition at line 344 of file vtss_qos_api.h.

6.139.2 Field Documentation

6.139.2.1 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.

6.139.2.2 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.

6.139.2.3 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.

6.139.2.4 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.

6.139.2.5 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.

6.139.2.6 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.

6.139.2.7 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.

6.139.2.8 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.

6.139.2.9 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.

6.139.2.10 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.

6.139.2.11 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.

6.139.2.12 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.

6.139.2.13 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.

6.139.2.14 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.

6.139.2.15 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.

6.139.2.16 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.

6.139.2.17 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.

6.139.2.18 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.

6.139.2.19 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.

6.139.2.20 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.

6.139.2.21 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.

6.139.2.22 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.

6.139.2.23 dwrr_enable

```
BOOL vtss_qos_port_conf_t::dwrr_enable
```

Enable Weighted fairness queueing

Definition at line 400 of file vtss_qos_api.h.

6.139.2.24 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.

6.139.2.25 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

6.140 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`

6.140.1 Detailed Description

Structure for Get PHY RC-PLL status.

Definition at line 1742 of file `vtss_phy_api.h`.

6.140.2 Field Documentation

6.140.2.1 `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`.

6.140.2.2 `cal_error`

```
u8 vtss_rcpll_status_t::cal_error
```

Calibration Error indication

Definition at line 1744 of file `vtss_phy_api.h`.

6.140.2.3 `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`

6.141 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`

6.141.1 Detailed Description

Restart status.

Definition at line 608 of file vtss_init_api.h.

6.141.2 Field Documentation

6.141.2.1 restart

```
vtss_restart_t vtss_restart_status_t::restart
```

Previous restart mode

Definition at line 609 of file vtss_init_api.h.

6.141.2.2 prev_version

```
vtss_version_t vtss_restart_status_t::prev_version
```

Previous API version

Definition at line 610 of file vtss_init_api.h.

6.141.2.3 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

6.142 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](#)

6.142.1 Detailed Description

Routing entry.

Definition at line 868 of file types.h.

6.142.2 Field Documentation

6.142.2.1 type

```
vtss_routing_entry_type_t vtss_routing_entry_t::type
```

Type of route

Definition at line 871 of file types.h.

6.142.2.2 ipv4_uc

`vtss_ipv4_uc_t` `vtss_routing_entry_t::ipv4_uc`

IPv6 unicast route

Definition at line 875 of file types.h.

6.142.2.3 ipv6_uc

`vtss_ipv6_uc_t` `vtss_routing_entry_t::ipv6_uc`

IPv6 unicast route

Definition at line 878 of file types.h.

6.142.2.4 route

`union { ... } vtss_routing_entry_t::route`

Route

6.142.2.5 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`

6.143 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]`

6.143.1 Detailed Description

Structure for Wake-On-LAN Secure-On Password.

Definition at line 1664 of file vtss_phy_api.h.

6.143.2 Field Documentation

6.143.2.1 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](#)

6.144 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](#)

6.144.1 Detailed Description

Serdes macro configuration.

Definition at line 363 of file vtss_init_api.h.

6.144.2 Field Documentation

6.144.2.1 serdes1g_vdd

`vtss_vdd_t` `vtss_serdes_macro_conf_t::serdes1g_vdd`

Serdes1g supply

Definition at line 364 of file `vtss_init_api.h`.

6.144.2.2 serdes6g_vdd

`vtss_vdd_t` `vtss_serdes_macro_conf_t::serdes6g_vdd`

Serdes6g supply

Definition at line 365 of file `vtss_init_api.h`.

6.144.2.3 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`.

6.144.2.4 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`

6.145 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`

6.145.1 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`.

6.145.2 Field Documentation

6.145.2.1 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`.

6.145.2.2 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`

6.146 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]`

6.146.1 Detailed Description

SGPIO configuration for a group.

Definition at line 797 of file vtss_misc_api.h.

6.146.2 Field Documentation

6.146.2.1 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.

6.146.2.2 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.

6.146.2.3 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

6.147 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]`

6.147.1 Detailed Description

SGPIO port configuration.

Definition at line 789 of file vtss_misc_api.h.

6.147.2 Field Documentation

6.147.2.1 enabled

```
BOOL vtss_sgpio_port_conf_t::enabled
```

Port enabled/disabled

Definition at line 791 of file vtss_misc_api.h.

6.147.2.2 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.

6.147.2.3 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

6.148 vtss_sgpi_port_data_t Struct Reference

SGPIO read data for a port.

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

Data Fields

- BOOL value [4]

6.148.1 Detailed Description

SGPIO read data for a port.

Definition at line 835 of file vtss_misc_api.h.

6.148.2 Field Documentation

6.148.2.1 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

6.149 vtss_shaper_t Struct Reference

Shaper.

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

Data Fields

- `vtss_burst_level_t level`
- `vtss_bitrate_t rate`

6.149.1 Detailed Description

Shaper.

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

6.149.2 Field Documentation

6.149.2.1 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`.

6.149.2.2 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`

6.150 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`

6.150.1 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`.

6.150.2 Field Documentation

6.150.2.1 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`.

6.150.2.2 squelsh

```
BOOL vtss_sync_clock_in_t::squelsh
```

Enable/disable of automatic squelsh

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

6.150.2.3 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`

6.151 `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`

6.151.1 Detailed Description

Struct containing configuration for a recovered clock output port.

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

6.151.2 Field Documentation

6.151.2.1 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`.

6.151.2.2 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`

6.152 `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`

6.152.1 Detailed Description

Tag Control Information (according to IEEE 802.1Q)

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

6.152.2 Field Documentation

6.152.2.1 vid

`vtss_vid_t vtss_tci_t::vid`

VLAN ID

Definition at line 208 of file `vtss_packet_api.h`.

6.152.2.2 cfi

`BOOL vtss_tci_t::cfi`

Canonical Format Indicator

Definition at line 209 of file vtss_packet_api.h.

6.152.2.3 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](#)

6.153 vtss_timeofday_t Struct Reference

Time of day structure.

```
#include <vtss_os_ecos.h>
```

Data Fields

- `u32 sec`
- `time_t sec`

6.153.1 Detailed Description

Time of day structure.

Definition at line 59 of file vtss_os_ecos.h.

6.153.2 Field Documentation

6.153.2.1 sec [1/2]

`u32 vtss_timeofday_t::sec`

Time of day in seconds

Definition at line 60 of file `vtss_os_ecos.h`.

6.153.2.2 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`

6.154 vtss_timestamp_t Struct Reference

Time stamp in seconds and nanoseconds.

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

Data Fields

- `u16 sec_msb`
- `u32 seconds`
- `u32 nanoseconds`

6.154.1 Detailed Description

Time stamp in seconds and nanoseconds.

Definition at line 1212 of file `types.h`.

6.154.2 Field Documentation

6.154.2.1 sec_msb

`u16 vtss_timestamp_t::sec_msb`

Seconds msb

Definition at line 1213 of file types.h.

6.154.2.2 seconds

`u32 vtss_timestamp_t::seconds`

Seconds

Definition at line 1214 of file types.h.

6.154.2.3 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`

6.155 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]`

6.155.1 Detailed Description

Trace group configuration.

Definition at line 97 of file vtss_misc_api.h.

6.155.2 Field Documentation

6.155.2.1 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](#)

6.156 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](#)

6.156.1 Detailed Description

external clock output configuration.

Definition at line 289 of file vtss_ts_api.h.

6.156.2 Field Documentation

6.156.2.1 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.

6.156.2.2 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.

6.156.2.3 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`

6.157 vtss_ts_id_t Struct Reference

Timestamp identifier.

```
#include <vtss_ts_api.h>
```

Data Fields

- `u32 ts_id`

6.157.1 Detailed Description

Timestamp identifier.

Definition at line 527 of file vtss_ts_api.h.

6.157.2 Field Documentation

6.157.2.1 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](#)

6.158 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](#)

6.158.1 Detailed Description

Hardware timestamping format mode for internal ports.

Definition at line 498 of file vtss_ts_api.h.

6.158.2 Field Documentation

6.158.2.1 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](#)

6.159 vtss_ts_operation_mode_t Struct Reference

Timestamp operation.

```
#include <vtss_ts_api.h>
```

Data Fields

- [vtss_ts_mode_t mode](#)

6.159.1 Detailed Description

Timestamp operation.

Definition at line 451 of file vtss_ts_api.h.

6.159.2 Field Documentation

6.159.2.1 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](#)

6.160 vtss_ts_timestamp_alloc_t Struct Reference

Timestamp allocation.

```
#include <vtss_ts_api.h>
```

Data Fields

- [u64 port_mask](#)
- [void * context](#)
- [void\(* cb \)\(void *context, u32 port_no, vtss_ts_timestamp_t *ts\)](#)

6.160.1 Detailed Description

Timestamp allocation.

Definition at line 613 of file vtss_ts_api.h.

6.160.2 Field Documentation

6.160.2.1 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.

6.160.2.2 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.

6.160.2.3 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](#)

6.161 vtss_ts_timestamp_t Struct Reference

Timestamp structure.

```
#include <vtss_ts_api.h>
```

Data Fields

- `u32 ts`
- `u32 id`
- `void * context`
- `BOOL ts_valid`

6.161.1 Detailed Description

Timestamp structure.

Definition at line 532 of file vtss_ts_api.h.

6.161.2 Field Documentation

6.161.2.1 ts

`u32 vtss_ts_timestamp_t::ts`

Timestamp value

Definition at line 533 of file vtss_ts_api.h.

6.161.2.2 id

`u32 vtss_ts_timestamp_t::id`

Timestamp identifier

Definition at line 534 of file vtss_ts_api.h.

6.161.2.3 context

`void* vtss_ts_timestamp_t::context`

Application specific context

Definition at line 535 of file vtss_ts_api.h.

6.161.2.4 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`

6.162 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`

6.162.1 Detailed Description

VCAP IPv4 address value and mask.

Definition at line 968 of file `types.h`.

6.162.2 Field Documentation

6.162.2.1 value

`vtss_ip_t vtss_vcap_ip_t::value`

Value

Definition at line 970 of file `types.h`.

6.162.2.2 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`

6.163 vtss_vcap_u128_t Struct Reference

VCAP 128 bit value and mask.

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

Data Fields

- `u8 value [16]`
- `u8 mask [16]`

6.163.1 Detailed Description

VCAP 128 bit value and mask.

Definition at line 954 of file types.h.

6.163.2 Field Documentation

6.163.2.1 value

`u8 vtss_vcap_u128_t::value[16]`

Value

Definition at line 956 of file types.h.

6.163.2.2 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`

6.164 vtss_vcap_u16_t Struct Reference

VCAP 16 bit value and mask.

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

Data Fields

- `u8 value [2]`
- `u8 mask [2]`

6.164.1 Detailed Description

VCAP 16 bit value and mask.

Definition at line 919 of file types.h.

6.164.2 Field Documentation

6.164.2.1 value

`u8 vtss_vcap_u16_t::value[2]`

Value

Definition at line 921 of file types.h.

6.164.2.2 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](#)

6.165 vtss_vcap_u24_t Struct Reference

VCAP 24 bit value and mask.

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

Data Fields

- [u8 value \[3\]](#)
- [u8 mask \[3\]](#)

6.165.1 Detailed Description

VCAP 24 bit value and mask.

Definition at line 926 of file types.h.

6.165.2 Field Documentation

6.165.2.1 value

`u8 vtss_vcap_u24_t::value[3]`

Value

Definition at line 928 of file types.h.

6.165.2.2 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](#)

6.166 vtss_vcap_u32_t Struct Reference

VCAP 32 bit value and mask.

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

Data Fields

- [u8 value \[4\]](#)
- [u8 mask \[4\]](#)

6.166.1 Detailed Description

VCAP 32 bit value and mask.

Definition at line 933 of file types.h.

6.166.2 Field Documentation

6.166.2.1 value

`u8 vtss_vcap_u32_t::value[4]`

Value

Definition at line 935 of file types.h.

6.166.2.2 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](#)

6.167 vtss_vcap_u40_t Struct Reference

VCAP 40 bit value and mask.

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

Data Fields

- [u8 value \[5\]](#)
- [u8 mask \[5\]](#)

6.167.1 Detailed Description

VCAP 40 bit value and mask.

Definition at line 940 of file types.h.

6.167.2 Field Documentation

6.167.2.1 value

`u8 vtss_vcap_u40_t::value[5]`

Value

Definition at line 942 of file types.h.

6.167.2.2 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`

6.168 vtss_vcap_u48_t Struct Reference

VCAP 48 bit value and mask.

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

Data Fields

- `u8 value [6]`
- `u8 mask [6]`

6.168.1 Detailed Description

VCAP 48 bit value and mask.

Definition at line 947 of file types.h.

6.168.2 Field Documentation

6.168.2.1 value

`u8 vtss_vcap_u48_t::value [6]`

Value

Definition at line 949 of file types.h.

6.168.2.2 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](#)

6.169 vtss_vcap_u8_t Struct Reference

VCAP 8 bit value and mask.

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

Data Fields

- [u8 value](#)
- [u8 mask](#)

6.169.1 Detailed Description

VCAP 8 bit value and mask.

Definition at line 912 of file types.h.

6.169.2 Field Documentation

6.169.2.1 value

`u8 vtss_vcap_u8_t::value`

Value

Definition at line 914 of file types.h.

6.169.2.2 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](#)

6.170 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](#)

6.170.1 Detailed Description

VCAP UDP/TCP port range.

Definition at line 975 of file types.h.

6.170.2 Field Documentation

6.170.2.1 in_range

`BOOL vtss_vcap_udp_tcp_t::in_range`

Port in range match

Definition at line 977 of file types.h.

6.170.2.2 low

`vtss_udp_tcp_t vtss_vcap_udp_tcp_t::low`

Port low value

Definition at line 978 of file types.h.

6.170.2.3 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`

6.171 vtss_vcap_vid_t Struct Reference

VCAP VLAN ID value and mask.

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

Data Fields

- `u16 value`
- `u16 mask`

6.171.1 Detailed Description

VCAP VLAN ID value and mask.

Definition at line 961 of file types.h.

6.171.2 Field Documentation

6.171.2.1 value

`u16 vtss_vcap_vid_t::value`

Value

Definition at line 963 of file types.h.

6.171.2.2 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`

6.172 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`

6.172.1 Detailed Description

VCAP universal value or range.

Definition at line 994 of file types.h.

6.172.2 Field Documentation

6.172.2.1 type

`vtss_vcap_vr_type_t vtss_vcap_vr_t::type`

Type

Definition at line 996 of file types.h.

6.172.2.2 value

`vtss_vcap_vr_value_t vtss_vcap_vr_t::value`

Value

Definition at line 1001 of file types.h.

6.172.2.3 mask

`vtss_vcap_vr_value_t vtss_vcap_vr_t::mask`

Mask, cleared bits are wildcards

Definition at line 1002 of file types.h.

6.172.2.4 v

`struct { ... } vtss_vcap_vr_t::v`

`type == VTSS_VCAP_VR_TYPE_VALUE_MASK`

6.172.2.5 low

`vtss_vcap_vr_value_t vtss_vcap_vr_t::low`

Low value

Definition at line 1006 of file types.h.

6.172.2.6 high

`vtss_vcap_vr_value_t vtss_vcap_vr_t::high`

High value

Definition at line 1007 of file types.h.

6.172.2.7 r

```
struct { ... } vtss_vcap_vr_t::r  
type == VTSS_VCAP_VR_TYPE_RANGE_XXXXXX
```

6.172.2.8 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`

6.173 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`

6.173.1 Detailed Description

VCE Action.

Definition at line 1006 of file vtss_l2_api.h.

6.173.2 Field Documentation

6.173.2.1 vid

`vtss_vid_t` `vtss_vce_action_t::vid`

Classified VLAN ID

Definition at line 1008 of file `vtss_l2_api.h`.

6.173.2.2 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`

6.174 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`

6.174.1 Detailed Description

Frame data for VTSS_VCE_TYPE_ETYPE.

Definition at line 948 of file `vtss_l2_api.h`.

6.174.2 Field Documentation

6.174.2.1 etype

`vtss_vcap_u16_t vtss_vce_frame_etype_t::etype`

Ethernet Type value

Definition at line 950 of file `vtss_l2_api.h`.

6.174.2.2 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`

6.175 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`

6.175.1 Detailed Description

Frame data for VTSS_VCE_TYPE_IPV4.

Definition at line 967 of file `vtss_l2_api.h`.

6.175.2 Field Documentation

6.175.2.1 fragment

`vtss_vcap_bit_t` vtss_vce_frame_ipv4_t::fragment

Fragment

Definition at line 969 of file vtss_l2_api.h.

6.175.2.2 options

`vtss_vcap_bit_t` vtss_vce_frame_ipv4_t::options

Header options

Definition at line 970 of file vtss_l2_api.h.

6.175.2.3 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.

6.175.2.4 proto

`vtss_vcap_u8_t` vtss_vce_frame_ipv4_t::proto

Protocol

Definition at line 972 of file vtss_l2_api.h.

6.175.2.5 sip

`vtss_vcap_ip_t` vtss_vce_frame_ipv4_t::sip

Source IP address

Definition at line 973 of file vtss_l2_api.h.

6.175.2.6 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`

6.176 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`

6.176.1 Detailed Description

Frame data for VTSS_VCE_TYPE_IPV6.

Definition at line 978 of file vtss_l2_api.h.

6.176.2 Field Documentation

6.176.2.1 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.

6.176.2.2 proto

`vtss_vcap_u8_t` `vtss_vce_frame_ipv6_t::proto`

Protocol

Definition at line 981 of file vtss_l2_api.h.

6.176.2.3 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.

6.176.2.4 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`

6.177 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`

6.177.1 Detailed Description

Frame data for VTSS_VCE_TYPE LLC.

Definition at line 955 of file vtss_l2_api.h.

6.177.2 Field Documentation

6.177.2.1 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`

6.178 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`

6.178.1 Detailed Description

Frame data for VTSS_VCE_TYPE_SNAP.

Definition at line 961 of file vtss_l2_api.h.

6.178.2 Field Documentation

6.178.2.1 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`

6.179 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`}

6.179.1 Detailed Description

VCE Key.

Definition at line 987 of file vtss_l2_api.h.

6.179.2 Field Documentation

6.179.2.1 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.

6.179.2.2 mac

```
vtss_vce_mac_t vtss_vce_key_t::mac
```

MAC header

Definition at line 990 of file vtss_l2_api.h.

6.179.2.3 tag

`vtss_vce_tag_t` `vtss_vce_key_t::tag`

Tag

Definition at line 991 of file vtss_l2_api.h.

6.179.2.4 type

`vtss_vce_type_t` `vtss_vce_key_t::type`

VCE frame type

Definition at line 992 of file vtss_l2_api.h.

6.179.2.5 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.

6.179.2.6 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.

6.179.2.7 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.

6.179.2.8 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`.

6.179.2.9 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`.

6.179.2.10 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`

6.180 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`

6.180.1 Detailed Description

VCE MAC header information.

Definition at line 930 of file `vtss_l2_api.h`.

6.180.2 Field Documentation

6.180.2.1 dmac_mc

`vtss_vcap_bit_t` `vtss_vce_mac_t::dmac_mc`

Multicast DMAC

Definition at line 932 of file `vtss_l2_api.h`.

6.180.2.2 dmac_bc

`vtss_vcap_bit_t` `vtss_vce_mac_t::dmac_bc`

Broadcast DMAC

Definition at line 933 of file `vtss_l2_api.h`.

6.180.2.3 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`

6.181 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`

6.181.1 Detailed Description

VLAN Control Entry.

Definition at line 1013 of file vtss_l2_api.h.

6.181.2 Field Documentation

6.181.2.1 id

`vtss_vce_id_t` `vtss_vce_t::id`

VCE ID

Definition at line 1015 of file vtss_l2_api.h.

6.181.2.2 key

`vtss_vce_key_t` `vtss_vce_t::key`

VCE Key

Definition at line 1016 of file vtss_l2_api.h.

6.181.2.3 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](#)

6.182 **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`

6.182.1 Detailed Description

VCE tag information.

Definition at line 938 of file vtss_l2_api.h.

6.182.2 Field Documentation

6.182.2.1 vid

`vtss_vcap_vid_t vtss_vce_tag_t::vid`

VLAN ID (12 bit)

Definition at line 940 of file vtss_l2_api.h.

6.182.2.2 pcp

`vtss_vcap_u8_t vtss_vce_tag_t::pcp`

PCP (3 bit)

Definition at line 941 of file vtss_l2_api.h.

6.182.2.3 dei

`vtss_vcap_bit_t vtss_vce_tag_t::dei`

DEI

Definition at line 942 of file vtss_l2_api.h.

6.182.2.4 tagged

`vtss_vcap_bit_t` `vtss_vce_tag_t::tagged`

Tagged/untagged frame

Definition at line 943 of file `vtss_l2_api.h`.

6.182.2.5 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`

6.183 vtss_vcl_port_conf_t Struct Reference

VCL port configuration.

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

Data Fields

- `BOOL dmac_dip`

6.183.1 Detailed Description

VCL port configuration.

Definition at line 878 of file `vtss_l2_api.h`.

6.183.2 Field Documentation

6.183.2.1 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`

6.184 vtss_vid_mac_t Struct Reference

MAC Address in specific VLAN.

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

Data Fields

- `vtss_vid_t vid`
- `vtss_mac_t mac`

6.184.1 Detailed Description

MAC Address in specific VLAN.

Definition at line 652 of file `types.h`.

6.184.2 Field Documentation

6.184.2.1 vid

`vtss_vid_t vtss_vid_mac_t::vid`

VLAN ID

Definition at line 654 of file `types.h`.

6.184.2.2 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](#)

6.185 vtss_vlan_conf_t Struct Reference

VLAN configuration.

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

Data Fields

- [vtss_etype_t s_etype](#)

6.185.1 Detailed Description

VLAN configuration.

Definition at line 625 of file vtss_l2_api.h.

6.185.2 Field Documentation

6.185.2.1 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](#)

6.186 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`

6.186.1 Detailed Description

VLAN port configuration.

Definition at line 664 of file `vtss_l2_api.h`.

6.186.2 Field Documentation

6.186.2.1 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`.

6.186.2.2 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`.

6.186.2.3 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.

6.186.2.4 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.

6.186.2.5 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](#)

6.187 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`

6.187.1 Detailed Description

VLAN tag with "arbitrary" TPID.

Definition at line 632 of file types.h.

6.187.2 Field Documentation

6.187.2.1 tpid

`vtss_etype_t` `vtss_vlan_tag_t::tpid`

Tag Protocol Identifier

Definition at line 633 of file types.h.

6.187.2.2 pcp

`vtss_tagprio_t` `vtss_vlan_tag_t::pcp`

Priority Code Point

Definition at line 634 of file types.h.

6.187.2.3 dei

`BOOL` `vtss_vlan_tag_t::dei`

Drop Eligible Indicator

Definition at line 635 of file types.h.

6.187.2.4 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`

6.188 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`

6.188.1 Detailed Description

VLAN translation group-to-VLAN configuration.

Definition at line 1104 of file `vtss_l2_api.h`.

6.188.2 Field Documentation

6.188.2.1 group_id

```
u16 vtss_vlan_trans_grp2vlan_conf_t::group_id
```

Group ID

Definition at line 1105 of file `vtss_l2_api.h`.

6.188.2.2 vid

```
vtss_vid_t vtss_vlan_trans_grp2vlan_conf_t::vid
```

VLAN ID

Definition at line 1106 of file `vtss_l2_api.h`.

6.188.2.3 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`

6.189 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]`

6.189.1 Detailed Description

VLAN translation port-to-group configuration.

Definition at line 1098 of file `vtss_l2_api.h`.

6.189.2 Field Documentation

6.189.2.1 group_id

`u16 vtss_vlan_trans_port2grp_conf_t::group_id`

Group ID

Definition at line 1099 of file `vtss_l2_api.h`.

6.189.2.2 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

6.190 vtss_vlan_vid_conf_t Struct Reference

VLAN ID configuration.

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

Data Fields

- `BOOL learning`
- `BOOL mirror`

6.190.1 Detailed Description

VLAN ID configuration.

Definition at line 740 of file vtss_l2_api.h.

6.190.2 Field Documentation

6.190.2.1 learning

```
BOOL vtss_vlan_vid_conf_t::learning
```

Enable/disable learning

Definition at line 742 of file vtss_l2_api.h.

6.190.2.2 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`

6.191 `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]`

6.191.1 Detailed Description

Structure for Wake-On-LAN MAC Address.

Definition at line 1656 of file `vtss_phy_api.h`.

6.191.2 Field Documentation

6.191.2.1 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_SYNCE
- #define VTSS_FEATURE_NPI
- #define VTSS_FEATURE_IRQ_CONTROL
- #define VTSS_OPT_VCORE_III 1
- #define VTSS_FEATURE_FDMA
- #define VTSS_FEATURE_AFI_FDMA
- #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_FDMA_IRQ_CONTEXT 0
- #define VTSS_OPT_FDMA_DEBUG 0
- #define VTSS_OPT_VAUI_EQ_CTRL 6
- #define VTSS_OPT_PORT_COUNT 0
- #define VTSS_PHY_OPT_VERIPH 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_OPT_VCORE_III

```
#define VTSS_OPT_VCORE_III 1
```

Internal VCORE-III (MIPS) CPU enabled by default

Definition at line 392 of file options.h.

7.2.2.62 VTSS_FEATURE_FDMA

```
#define VTSS_FEATURE_FDMA
```

Frame DMA

Definition at line 395 of file options.h.

7.2.2.63 VTSS_FEATURE_AFI_FDMA

```
#define VTSS_FEATURE_AFI_FDMA
```

FDMA-based Automatic Frame injector supported on EVC-enabled parts

Definition at line 397 of file options.h.

7.2.2.64 VTSS_FEATURE_LED_POW_REDUC

```
#define VTSS_FEATURE_LED_POW_REDUC
```

LED power reduction

Definition at line 400 of file options.h.

7.2.2.65 VTSS_FEATURE_INTERRUPTS

```
#define VTSS_FEATURE_INTERRUPTS
```

Port Interrupt support

Definition at line 401 of file options.h.

7.2.2.66 VTSS_FEATURE_VLAN_TRANSLATION

```
#define VTSS_FEATURE_VLAN_TRANSLATION
```

VLAN Translation

Definition at line 402 of file options.h.

7.2.2.67 VTSS_FEATURE_SFLOW

```
#define VTSS_FEATURE_SFLOW
```

Statistical flow sampling

Definition at line 403 of file options.h.

7.2.2.68 VTSS_FEATURE_MIRROR_CPU

```
#define VTSS_FEATURE_MIRROR_CPU
```

CPU mirroring

Definition at line 404 of file options.h.

7.2.2.69 VTSS_FEATURE_SERDES_MACRO_SETTINGS

```
#define VTSS_FEATURE_SERDES_MACRO_SETTINGS
```

Hooks for Serdes Macro configuration

Definition at line 405 of file options.h.

7.2.2.70 VTSS_CHIP CU PHY

```
#define VTSS_CHIP CU PHY
```

Copper PHY chip

Definition at line 540 of file options.h.

7.2.2.71 VTSS_OPT_TRACE

```
#define VTSS_OPT_TRACE 1
```

Trace enabled by default

Definition at line 548 of file options.h.

7.2.2.72 VTSS_OPT_FDMA_IRQ_CONTEXT

```
#define VTSS_OPT_FDMA_IRQ_CONTEXT 0
```

Deferred interrupt context by default Use of VTSS_OPT_FDMA_VER is the preferred way to indicate which version of the FDMA API is required Make sure noone uses this one anymore

Definition at line 577 of file options.h.

7.2.2.73 VTSS_OPT_FDMA_DEBUG

```
#define VTSS_OPT_FDMA_DEBUG 0
```

FDMA debug disabled by default

Definition at line 599 of file options.h.

7.2.2.74 VTSS_OPT_VAUI_EQ_CTRL

```
#define VTSS_OPT_VAUI_EQ_CTRL 6
```

Default equalization control

Definition at line 605 of file options.h.

7.2.2.75 VTSS_OPT_PORT_COUNT

```
#define VTSS_OPT_PORT_COUNT 0
```

Use all target ports by default

Definition at line 609 of file options.h.

7.2.2.76 VTSS_PHY_OPT_VERIPHY

```
#define VTSS_PHY_OPT_VERIPHY 1
```

VeriPHY enabled by default

Definition at line 613 of file options.h.

7.2.2.77 VTSS_FEATURE_WARM_START

```
#define VTSS_FEATURE_WARM_START
```

Warm start

Definition at line 617 of file options.h.

7.2.2.78 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 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_VERIPHY_STATUS_OK = 0, VTSS_VERIPHY_STATUS_OPEN = 1, VTSS_VERIPHY_STATUS_SHORT = 2, VTSS_VERIPHY_STATUS_ABNORM = 4, VTSS_VERIPHY_STATUS_SHORT_A = 8, VTSS_VERIPHY_STATUS_SHORT_B = 9, VTSS_VERIPHY_STATUS_SHORT_C = 10, VTSS_VERIPHY_STATUS_SHORT_D = 11, VTSS_VERIPHY_STATUS_COUPL_A = 12, VTSS_VERIPHY_STATUS_COUPL_B = 13, VTSS_VERIPHY_STATUS_COUPL_C = 14, VTSS_VERIPHY_STATUS_COUPL_D = 15, VTSS_VERIPHY_STATUS_UNKNOWN = 16, VTSS_VERIPHY_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

<code>VTSS_SPEED_FIBER_NOT_SUPPORTED_OR_DISABLED</code>	Fiber not supported/ Fiber port disabled
<code>VTSS_SPEED_FIBER_100FX</code>	100BASE-FX
<code>VTSS_SPEED_FIBER_1000X</code>	1000BASE-X
<code>VTSS_SPEED_FIBER_AUTO</code>	Auto detection
<code>VTSS_SPEED_FIBER_DISABLED</code>	Obsolete - use <code>VTSS_SPEED_FIBER_NOT_SUPPORTED_OR_DISABLED</code> instead

Definition at line 255 of file port.h.

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`
 - IPv4 unicast routing entry.*
- struct `vtss_routing_entry_t`
 - IPv6 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 `PRIu64` "llu"
- #define `PRIi64` "lli"
- #define `PRIx64` "llx"
- #define `VTSS_BIT64`(x) (1ULL << (x))
- #define `VTSS_BITMASK64`(x) ((1ULL << (x)) - 1)
- #define `VTSS_EXTRACT_BITFIELD64`(x, o, w) (((x) >> (o)) & `VTSS_BITMASK64`(w))
- #define `VTSS_ENCODE_BITFIELD64`(x, o, w) (((u64)(x) & `VTSS_BITMASK64`(w)) << (o))
- #define `VTSS_ENCODE_BITMASK64`(o, w) (`VTSS_BITMASK64`(w) << (o))
- #define `TRUE` 1
- #define `FALSE` 0
- #define `VTSS_PACKET_RATE_DISABLED` 0xffffffff
- #define `VTSS_PORT_COUNT` 1
- #define `VTSS_PORT_COUNT` 26
- #define `VTSS_PORTS` `VTSS_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_MACADDR** = -67, **VTSS_RC_ERR_MACSEC_COULD_NOT_PRG_SA_FLOW** = -68, **VTSS_RC_ERR_MACSEC_COULD_NOT_ENA_SA** = -69, **VTSS_RC_ERR_MACSEC_COULD_NOT_SET_SA** = -70, **VTSS_RC_ERR_MACSEC_INVALID_BYPASS_HDR_LEN** = -71, **VTSS_RC_ERR_MACSEC_SC_NOT_FOUND** = -72, **VTSS_RC_ERR_MACSEC_NO_CTRL_FRM_MATCH** = -73, **VTSS_RC_ERR_MACSEC_COULD_NOT_SET_PATTERN** = -74, **VTSS_RC_ERR_MACSEC_TIMEOUT_ISSUE** = -75, **VTSS_RC_ERR_MACSEC_COULD_NOT_EMPTY_EGRESS** = -76, **VTSS_RC_ERR_MACSEC_AN_NOT_CREATED** = -77, **VTSS_RC_ERR_MACSEC_COULD_NOT_EMPTY_INGRESS** = -78, **VTSS_RC_ERR_MACSEC_TX_SC_NOT_EXIST** = -80, **VTSS_RC_ERR_MACSEC_COULD_NOT_DISABLE_SA** = -81, **VTSS_RC_ERR_MACSEC_COULD_NOT_DEL_RX_SA** = -82, **VTSS_RC_ERR_MACSEC_COULD_NOT_DEL_TX_SA** = -83, **VTSS_RC_ERR_MACSEC_PATTERN_NOT_SET** = -84, **VTSS_RC_ERR_MACSEC_HW_RESOURCE_EXHUSTED** = -85, **VTSS_RC_ERR_MACSEC_SCI_ALREADY_EXISTS** = -86, **VTSS_RC_ERR_MACSEC_SC_RESOURCE_NOT_FOUND** = -87, **VTSS_RC_ERR_MACSEC_RX_AN_ALREADY_IN_USE** = -88, **VTSS_RC_ERR_MACSEC_EMPTY_RECORD** = -89, **VTSS_RC_ERR_MACSEC_COULD_NOT_PRG_XFORM**

```
= -90, VTSS_RC_ERR_MACSEC_COULD_NOT_TOGGLE_SA = -91, VTSS_RC_ERR_MACSEC_TX_AN_ALREADY_IN_USE
= -92,
VTSS_RC_ERR_MACSEC_ALL_AVAILABLE_SA_IN_USE = -93, VTSS_RC_ERR_MACSEC_MATCH_DISABLE
= -94, VTSS_RC_ERR_MACSEC_ALL_CP_RULES_IN_USE = -95, VTSS_RC_ERR_MACSEC_PATTERN_PRIO_NOT_VALID
= -96,
VTSS_RC_ERR_MACSEC_BUFFER_TOO_SMALL = -97, VTSS_RC_ERR_MACSEC_FRAME_TOO_LONG
= -98, VTSS_RC_ERR_MACSEC_FRAME_TRUNCATED = -99, VTSS_RC_ERR_MACSEC_PHY_POWERED_DOWN
= -100,
VTSS_RC_ERR_MACSEC_PHY_NOT_MACSEC_CAPABLE = -101, VTSS_RC_ERR_MACSEC_AN_NOT_EXIST
= -102, VTSS_RC_ERR_MACSEC_NO_PATTERN_CFG = -103, VTSS_RC_ERR_MACSEC_MAX_MTU =
-105,
VTSS_RC_ERR_MACSEC_UNEXPECT_CP_MODE = -106, VTSS_RC_ERR_MACSEC_COULD_NOT_DISABLE_AN
= -107, VTSS_RC_ERR_MACSEC_RULE_OUT_OF_RANGE = -108, VTSS_RC_ERR_MACSEC_RULE_NOT_EXIST
= -109,
VTSS_RC_ERR_MACSEC_CSR_READ = -110, VTSS_RC_ERR_MACSEC_CSR_WRITE = -111,
VTSS_RC_ERR_PHY_6G_RCPLL_ON_BASE_PORT_ONLY = -112, VTSS_RC_ERR_INVALID_NULL_PTR
= -200 }
```

Error codes.

- enum `vtss_mem_flags_t` { `VTSS_MEM_FLAGS_NONE` = 0x0, `VTSS_MEM_FLAGS_DMA` = 0x1, `VTSS_MEM_FLAGS_PERSIST` = 0x2 }

Memory allocation flags.

- enum `vtss_port_interface_t` {
`VTSS_PORT_INTERFACE_NO_CONNECTION`, `VTSS_PORT_INTERFACE_LOOPBACK`, `VTSS_PORT_INTERFACE_INTERNAL`,
`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_PORT_COUNT
```

Number of ports

Definition at line 444 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 VTSSETYPE_VTSS

```
#define VTSSETYPE_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_FRM	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_HDRLEN	Invalid header bypass length
VTSS_RC_ERR_MACSEC_SC_NOT_FOUND	Could not find SC (from sci)
VTSS_RC_ERR_MACSEC_NO_CTRL_FRM_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
```

Serdes macro mode.

Enumerator

VTSS_SERDES_MODE_DISABLE	Disable serdes
VTSS_SERDES_MODE_XAUI_12G	XAUI 12G mode
VTSS_SERDES_MODE_XAUI	XAUI 10G mode
VTSS_SERDES_MODE_RXAUI	RXAUI 10G mode
VTSS_SERDES_MODE_RXAUI_12G	RXAUI 12G mode
VTSS_SERDES_MODE_2G5	2.5G mode
VTSS_SERDES_MODE_QSGMII	QSGMII mode
VTSS_SERDES_MODE_SGMII	SGMII mode
VTSS_SERDES_MODE_100FX	100FX mode
VTSS_SERDES_MODE_1000BaseX	1000BaseX mode
VTSS_SERDES_MODE_SFI	LAN/10G mode
VTSS_SERDES_MODE_SFI_DAC	LAN/10G DAC(CU)
VTSS_SERDES_MODE_IDLE	Send idles

Definition at line 490 of file types.h.

7.5.4.5 vtss_storm_policer_mode_t

enum `vtss_storm_policer_mode_t`

Storm policer mode configuration.

Enumerator

VTSS_STORM_POLICER_MODE_PORTS_AND_CPU	Police both CPU and front port destinations
VTSS_STORM_POLICER_MODE_PORTS_ONLY	Police front port destinations only
VTSS_STORM_POLICER_MODE_CPU_ONLY	Police CPU destination only

Definition at line 572 of file types.h.

7.5.4.6 vtss_policer_type_t

enum `vtss_policer_type_t`

Dual leaky buckets policer configuration.

Enumerator

VTSS_POLICER_TYPE_MEF	MEF bandwidth profile
VTSS_POLICER_TYPE_SINGLE	Single bucket policer (CIR/CBS)

Definition at line 587 of file types.h.

7.5.4.7 vtss_vlan_frame_t

enum `vtss_vlan_frame_t`

VLAN acceptable frame type.

Enumerator

VTSS_VLAN_FRAME_ALL	Accept all frames
VTSS_VLAN_FRAME_TAGGED	Accept tagged frames only
VTSS_VLAN_FRAME_UNTAGGED	Accept untagged frames only

Definition at line 618 of file types.h.

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_fdma_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](#)

- 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,
    const 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>
```

Data Structures

- struct [tag_vtss_fdma_list](#)
Software DCB structure.
- struct [vtss_fdma_tx_info_t](#)

- struct `vtss_fdma_ch_cfg_t`
Channel configuration structure.
- struct `vtss_fdma_cfg_t`
FDMA configuration structure.
- struct `vtss_fdma_throttle_cfg_t`

Macros

- `#define VTSS_FDMA_CH_CNT 8`
- `#define VTSS_PHYS_PORT_CNT 26`
- `#define VTSS_FDMA_DCB_SIZE_BYTES 24`
- `#define VTSS_FDMA_HDR_SIZE_BYTES 16`
- `#define VTSS_FDMA_MAX_DATA_PER_DCB_BYTES 16380`
- `#define VTSS_FDMA_MIN_FRAME_SIZE_BYTES 64`
- `#define VTSS_FDMA_MIN_DATA_PER_INJ_SOF_DCB_BYTES (VTSS_FDMA_HDR_SIZE_BYTES)`
- `#define VTSS_FDMA_MIN_DATA_PER_XTR_SOF_DCB_BYTES (VTSS_FDMA_HDR_SIZE_BYTES)`
- `#define VTSS_FDMA_MIN_DATA_PER_NON_SOF_DCB_BYTES 1`
- `#define VTSS_FDMA_MAX_FRAME_SIZE_BYTES 10000`
- `#define VTSS_OS_DCACHE_LINE_SIZE_BYTES 32`
- `#define VTSS_OS_COMPILER_ATTRIBUTE_ALIGNED(x) __attribute ((aligned(x)))`
- `#define VTSS_AFI_FPS_MAX (1000000)`
- `#define VTSS_FDMA_CCM_QUOTIENT_MAX 3`
- `#define VTSS_FDMA_CCM_FREQ_LIST_LEN 5`
- `#define VTSS_FDMA_CCM_FPS_MAX VTSS_AFI_FPS_MAX`

Typedefs

- `typedef i32 vtss_fdma_ch_t`
Frame DMA channel number. Channels are numbered in range [0; VTSS_FDMA_CH_CNT].
- `typedef struct tag_vtss_fdma_list vtss_fdma_list_t`
Software DCB structure.

Enumerations

- enum `vtss_fdma_afi_type_t` { `VTSS_FDMA_AFI_TYPE_AUTO` = 0, `VTSS_FDMA_AFI_TYPE_FDMA`, `VTSS_FDMA_AFI_TYPE_SWC` }
Select which AFI to use.
- enum `vtss_fdma_ch_usage_t` { `VTSS_FDMA_CH_USAGE_UNUSED`, `VTSS_FDMA_CH_USAGE_XTR`, `VTSS_FDMA_CH_USAGE_INJ`, `VTSS_FDMA_CH_USAGE_CCM` }
Channel usage.
- enum `vtss_fdma_dcb_type_t` { `VTSS_FDMA_DCB_TYPE_XTR`, `VTSS_FDMA_DCB_TYPE_INJ`, `VTSS_FDMA_DCB_TYPE_A` }
DCB type identifying a DCB.

Functions

- `vtss_rc vtss_fdma_uninit (const vtss_inst_t inst)`
Uninitialize FDMA.
- `vtss_rc vtss_fdma_cfg (const vtss_inst_t inst, const vtss_fdma_cfg_t *const cfg)`
Configure FDMA.
- `vtss_rc vtss_fdma_dcb_release (const vtss_inst_t inst, vtss_fdma_list_t *const list)`
Release DCBs.
- `vtss_rc vtss_fdma_tx (const vtss_inst_t inst, vtss_fdma_list_t *list, vtss_fdma_tx_info_t *const fdma_info, vtss_packet_tx_info_t *const tx_info)`
Inject a frame.
- `vtss_rc vtss_fdma_tx_info_init (const vtss_inst_t inst, vtss_fdma_tx_info_t *const fdma_info)`
Initialize a `vtss_fdma_tx_info_t` structure.
- `vtss_rc vtss_fdma_afi_cancel (const vtss_inst_t inst, const u8 *const frm_ptr)`
Cancel an ongoing AFI transmission.
- `vtss_rc vtss_fdma_afi_frm_cnt (const vtss_inst_t inst, const u8 *const frm_ptr, u64 *const frm_cnt)`
Get an interim frame count for a given periodically injected frame.
- `vtss_rc vtss_fdma_dcb_get (const vtss_inst_t inst, u32 dcb_cnt, vtss_fdma_dcb_type_t dcb_type, vtss_fdma_list_t **list)`
Get one or more DCBs suitable for frame injection.
- `vtss_rc vtss_fdma_throttle_cfg_get (const vtss_inst_t inst, vtss_fdma_throttle_cfg_t *const cfg)`
Get current throttle configuration.
- `vtss_rc vtss_fdma_throttle_cfg_set (const vtss_inst_t inst, const vtss_fdma_throttle_cfg_t *const cfg)`
Configure throttling.
- `vtss_rc vtss_fdma_throttle_tick (const vtss_inst_t inst)`
Generate throttle tick.
- `vtss_rc vtss_fdma_stats_clr (const vtss_inst_t inst)`
Clear FDMA statistics.
- `vtss_rc vtss_fdma_irq_handler (const vtss_inst_t inst, void *const ctxt)`
FDMA Interrupt Handler.

7.11.1 Detailed Description

Frame DMA API.

7.11.2 Macro Definition Documentation

7.11.2.1 VTSS_FDMA_CH_CNT

```
#define VTSS_FDMA_CH_CNT 8
```

Number of DMA Channels.

Definition at line 200 of file vtss_fdma_api.h.

7.11.2.2 VTSS_PHYS_PORT_CNT

```
#define VTSS_PHYS_PORT_CNT 26
```

Number of Port Modules, excluding CPU Port Module.

Definition at line 201 of file vtss_fdma_api.h.

7.11.2.3 VTSS_FDMA_DCB_SIZE_BYTES

```
#define VTSS_FDMA_DCB_SIZE_BYTES 24
```

Number of bytes in a DCB.

Definition at line 202 of file vtss_fdma_api.h.

7.11.2.4 VTSS_FDMA_HDR_SIZE_BYTES

```
#define VTSS_FDMA_HDR_SIZE_BYTES 16
```

Max(XTR_HDR_SZ, INJ_HDR_SZ). Worst-case is INJ (8 for IFH, vtss_fdma_inj_props_t::ptp_action != 0 => +4 for timestamp. @switch_frm == TRUE => +4 for VLAN tag)

Definition at line 203 of file vtss_fdma_api.h.

7.11.2.5 VTSS_FDMA_MAX_DATA_PER_DCB_BYTES

```
#define VTSS_FDMA_MAX_DATA_PER_DCB_BYTES 16380
```

For both injection and extraction, the maximum number of bytes that one single DCB's associated data area can refer to. If you need to inject or extract larger frames, use multiple DCBs.

Definition at line 299 of file vtss_fdma_api.h.

7.11.2.6 VTSS_FDMA_MIN_FRAME_SIZE_BYTES

```
#define VTSS_FDMA_MIN_FRAME_SIZE_BYTES 64
```

The minimum allowed frame size (excluding IFH and CMD fields, but including FCS) in bytes. This is defined for legacy reasons. The FDMA will automatically adjust any frame below the minimum ethernet size to the minimum ethernet size before transmission.

Definition at line 302 of file vtss_fdma_api.h.

7.11.2.7 VTSS_FDMA_MIN_DATA_PER_INJ_SOF_DCB_BYTES

```
#define VTSS_FDMA_MIN_DATA_PER_INJ_SOF_DCB_BYTES (VTSS_FDMA_HDR_SIZE_BYTES)
```

Defines the minimum size in bytes of an injection SOF-DCB's associated data area.

Definition at line 309 of file vtss_fdma_api.h.

7.11.2.8 VTSS_FDMA_MIN_DATA_PER_XTR_SOF_DCB_BYTES

```
#define VTSS_FDMA_MIN_DATA_PER_XTR_SOF_DCB_BYTES (VTSS_FDMA_HDR_SIZE_BYTES)
```

Defines the minimum size in bytes of an extraction SOF-DCB's associated data area. Since every DCB can become a SOF DCB, this value also defines the minimum size that the user must allocate per DCB.

Definition at line 310 of file vtss_fdma_api.h.

7.11.2.9 VTSS_FDMA_MIN_DATA_PER_NON_SOFR_DCB_BYTES

```
#define VTSS_FDMA_MIN_DATA_PER_NON_SOFR_DCB_BYTES 1
```

Defines the minimum size in bytes of an injection/extraction non-SOF- DCB's associated data area.

Definition at line 313 of file vtss_fdma_api.h.

7.11.2.10 VTSS_FDMA_MAX_FRAME_SIZE_BYTES

```
#define VTSS_FDMA_MAX_FRAME_SIZE_BYTES 10000
```

The maximum allowed total frame size (excluding IFH and CMD fields) in bytes.

Definition at line 314 of file vtss_fdma_api.h.

7.11.2.11 VTSS_OS_DCACHE_LINE_SIZE_BYTES

```
#define VTSS_OS_DCACHE_LINE_SIZE_BYTES 32
```

The number of bytes one DCache-line is made up of.

Definition at line 317 of file vtss_fdma_api.h.

7.11.2.12 VTSS_OS_COMPILER_ATTRIBUTE_ALIGNED

```
#define VTSS_OS_COMPILER_ATTRIBUTE_ALIGNED( x ) __attribute__ ((aligned(x)))
```

In some special cases, it is of utmost importance that a certain variable has a certain memory alignment. Applications for this is e.g. placing variables on cache-line boundaries.

Definition at line 321 of file vtss_fdma_api.h.

7.11.2.13 VTSS_AFI_FPS_MAX

```
#define VTSS_AFI_FPS_MAX (1000000)
```

Maximum number of frames to inject per second using FDMA-based AFI

Definition at line 929 of file vtss_fdma_api.h.

7.11.2.14 VTSS_FDMA_CCM_QUOTIENT_MAX

```
#define VTSS_FDMA_CCM_QUOTIENT_MAX 3
```

Maximum quotient allowed on a given AFI channel.

Definition at line 1751 of file vtss_fdma_api.h.

7.11.2.15 VTSS_FDMA_CCM_FREQ_LIST_LEN

```
#define VTSS_FDMA_CCM_FREQ_LIST_LEN 5
```

AFI:

The following defines the maximum number of different frequencies (all multiples of each other) that can run on the same channel.

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

Definition at line 1790 of file vtss_fdma_api.h.

7.11.2.16 VTSS_FDMA_CCM_FPS_MAX

```
#define VTSS_FDMA_CCM_FPS_MAX VTSS_AFI_FPS_MAX
```

AFI:

The following defines the maximum number of frames per second that can be used in vtss_fdma_inj_props_t's fps member. There is absolutely no guarantee that the actual H/W architecture can reach this number of frames per second.

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

Definition at line 1807 of file vtss_fdma_api.h.

7.11.3 Typedef Documentation

7.11.3.1 vtss_fdma_ch_t

```
typedef i32 vtss_fdma_ch_t
```

Frame DMA channel number. Channels are numbered in range [0; VTSS_FDMA_CH_CNT[.

FREQUENTLY ASKED QUESTIONS

Q: What CPUs does the FDMA work with?

A: The FDMA Driver Kit *must* execute on the embedded processor, since an external CPU doesn't have access to the FDMA silicon.

-----oOo-----

Q: Can the FDMA be used in parallel with "manual" frame transmission or reception?

A: It is not recommended to use the FDMA together with "manual" frame transmission or reception (manual refers to the fact that the CPU spends clock cycles to read or write every single byte to the relevant registers within the switch core). However, for frame transmission, it is possible to have the FDMA working on one set of front ports and the manual transmission working on another set. For frame reception, it is possible to have the FDMA extract from one set of the extraction queues, and have the manual reception working on another set. The sets must be disjunct.

-----oOo-----

Q: Are any of the API functions blocking?

A: No, none of the functions are blocking. When an API function needs exclusive access to a global variable, it uses a macro (either [VTSS_OS_INTERRUPT_DISABLE\(\)](#) or [VTSS_OS_SCHEDULER_LOCK\(\)](#), depending on the value of the VTSS_OPT_FDMA_IRQ_CONTEXT preprocessor symbol) to ask for OS-specific support to globally ensure mutual exclusiveness.

-----oOo-----

Q: Who allocates memory?

A: The API is built in such a way that it is up to the caller to allocate and free memory. Some may wish to allocate all the buffers needed by the FDMA statically, while others may want to allocate it dynamically. If the FDMA code was to implement code for all possible scenarios, it would not be as flexible as it is as of today.

TERMS AND ABBREVIATIONS

Extraction:

This term is used by the FDMA silicon and driver kit and is equivalent to what higher levels of software would call Rx, i.e. packet reception. Extraction is abbreviated XTR.

Injection:

This term is used by the FDMA silicon and driver kit and is equivalent to what higher levels of software would call Tx, i.e. packet transmission. Injection is abbreviated INJ.

Packet Module:

The software module that implements all the calls to the FDMA API.

Call context:

The context that a function must be called in. In a multithreaded environment this could be "thread" or "DSR".

Re-entrancy:

Some of the functions are allowed to be called simultaneously from several threads.

DSR:

Deferred Service Routine. This is the term that eCos uses for the context that is scheduled by an IRQ handler. The context is assumed to be interruptible only by another hardware interrupt, i.e. the scheduler cannot schedule other DSRs or threads while a DSR is running. In Linux, this is known as the "bottom half". A DSR can never wait for critical sections or semaphores or the like.

DCB:

In the FDMA driver context, a DCB is normally a software DCB, i.e. an instance of the `vtss_fdma_list_t` structure. Each software DCB embeds a hardware DCB, which is filled by the FDMA driver code and passed to the FDMA silicon, i.e. the higher levels of software need not be concerned about hardware DCBs, but they need to allocate room for them, which is therefore done in the software DCB. The same software DCB type is used for injection and extraction, but some of the fields' meaning differ for the two. Please refer to the `vtss_fdma_list_t` structure for the exact interpretation and Packet Module requirements for the DCBs.

SOF DCB:

Start-Of-Frame DCB. The first (software or hardware) DCB in a list of DCBs making up one frame. The list is at least 1 item long.

EOF DCB:

End-Of-Frame DCB. The last (software or hardware) DCB in a list of DCBs making up one frame. The list is at least 1 item long, so the EOF DCB may be identical to the SOF DCB.

Non-SOF DCBs:

All but the first DCB in a list of (software or hardware) DCBs making up one frame.

Extraction Queue:

The chip contains a number of queues for temporarily storing frames destined for the CPU. These so-called extraction queues are of configurable length and are located in the CPU Port Module. They share memory with the CPU Port Module's injection queue. The splitting of memory between the queues is not handled by the FDMA driver code. Likewise, the assignment of a particular type of frames to an extraction queue (which typically takes place in the chip's analyzer (ANA) is not done by the FDMA driver code. The valid range of extraction queue numbers is [VTSS_PACKET_RX_QUEUE_START; VTSS_PACKET_RX_QUEUE_END[.

DMA Channel:

A DMA channel is used per flow. For extraction each extraction queue is statically mapped to a DMA channel through the call to `vtss_fdma_xtr_cfg()`. For injection, there is no static mapping between a channel number and a front port number, but a channel must still be assigned for injection. Valid channel numbers are in the range [0; VTSS_FDMA_CH_CNT[. An intrinsic priority exists among the channels, in that higher-numbered channels have higher priority. Thus, if two channels serve two different extraction queues, and both report frames present, then the higher numbered channel (which is not necessarily the higher numbered extraction queue!) will get serviced first.

LAYOUT OF INJECTED AND EXTRACTED FRAMES

Injection:

When injecting a frame, the SOF DCB's data pointer must point to an area of at least VTSS_FDMA_INJ_HDR_SIZE_BYTES (VTSS_OPT_FDMA_VER == 1) or VTSS_FDMA_HDR_SIZE_BYTES (VTSS_OPT_FDMA_VER >= 2), which is reserved by the FDMA. The byte following these initial bytes must therefore be the first byte of the frame's DMAC. If VTSS_OPT_FDMA_VER >= 2, then the same buffers can be used for both injection and extraction, and the amount of data to reserve in the beginning of a frame is given by VTSS_FDMA_HDR_SIZE_BYTES.

Extraction:

When an extracted frame is delivered to the callback function, the first VTSS_FDMA_XTR_HDR_SIZE_BYTES (VTSS_OPT_FDMA_VER == 1) of the SOF DCB's data area contain the frame's IFH. For VTSS_OPT_FDMA_VER >= 2, the list->ifh_ptr points to the IFH, which may or may not (depending on architecture) be the same as the list->data member. The list->frm points to the actual frame data. If the frame contains a stack header, then it'll be stripped from the frame (for architectures that carry this in the payload rather than the IFH) and placed in the IFH prior to callback. The list->frm_ptr points to the actual frame data.

Definition at line 194 of file vtss_fdma_api.h.

7.11.3.2 vtss_fdma_list_t

```
typedef struct tag_vtss_fdma_list vtss_fdma_list_t
```

Software DCB structure.

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

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

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

7.11.4 Enumeration Type Documentation

7.11.4.1 vtss_fdma_afi_type_t

```
enum vtss_fdma_afi_type_t
```

Select which AFI to use.

On platforms that only support either an FDMA-based or a switch-core-based AFI, select the type it supports or VTSS_FDMA_AFI_TYPE_AUTO, which will resort to the one it supports.

On platforms that support both AFI types, selecting VTSS_FDMA_AFI_TYPE_AUTO will cause the FDMA to look at FDMA-only properties, that is, whether frame counting or sequence numbering is enabled, and if so choose the FDMA-based, otherwise it will choose the switch-core based.

Enumerator

VTSS_FDMA_AFI_TYPE_AUTO	The FDMA driver chooses an appropriate frame injector.
VTSS_FDMA_AFI_TYPE_FDMA	Tell FDMA driver to use the FDMA-based AFI.
VTSS_FDMA_AFI_TYPE_SWC	Tell FDMA driver to use the switch-core-based AFI.

Definition at line 731 of file vtss_fdma_api.h.

7.11.4.2 vtss_fdma_ch_usage_t

enum [vtss_fdma_ch_usage_t](#)

Channel usage.

A given FDMA channel can either be used for extraction or injection.

Enumerator

VTSS_FDMA_CH_USAGE_UNUSED	The channel is not currently in use.
VTSS_FDMA_CH_USAGE_XTR	The channel is used/supposed to be used for frame extraction.
VTSS_FDMA_CH_USAGE_INJ	The channel is used/supposed to be used for frame injection.
VTSS_FDMA_CH_USAGE_CCM	The channel is used/supposed to be used for period frame injection

Definition at line 1547 of file vtss_fdma_api.h.

7.11.4.3 vtss_fdma_dcb_type_t

enum [vtss_fdma_dcb_type_t](#)

DCB type identifying a DCB.

Enumerator

VTSS_FDMA_DCB_TYPE_XTR	The DCB is an extraction DCB. Not needed by application.
VTSS_FDMA_DCB_TYPE_INJ	The DCB is an injection DCB. Needed in call to vtss_fdma_dcb_get() .
VTSS_FDMA_DCB_TYPE_AFI	The DCB is an AFI DCB. Needed in call to vtss_fdma_dcb_get() .

Definition at line 2580 of file vtss_fdma_api.h.

7.11.5 Function Documentation

7.11.5.1 vtss_fdma_uninit()

```
vtss_rc vtss_fdma_uninit (
    const vtss_inst_t inst )
```

Uninitialize FDMA.

Rarely used. Use only if you want to make a controlled shut-down of the FDMA. Disables all FDMA channels and interrupts, and clears pending interrupts.

- Call context:
Thread
- Re-entrant:
No

Parameters

<i>inst</i>	[IN]: Target instance reference.
-------------	----------------------------------

Returns

VTSS_RC_OK on success.
VTSS_RC_ERROR if a supplied parameter was erroneous.

7.11.5.2 vtss_fdma_cfg()

```
vtss_rc vtss_fdma_cfg (
    const vtss_inst_t inst,
    const vtss_fdma_cfg_t *const cfg )
```

Configure FDMA.

Call this function before any other FDMA API function.

- Call context:
Thread
- Re-entrant:
No

Parameters

<i>inst</i>	[IN]: Target instance reference.
<i>cfg</i>	[IN]: FDMA configuration.

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR if a supplied parameter was erroneous.

7.11.5.3 vtss_fdma_dcb_release()

```
vtss_rc vtss_fdma_dcb_release (
    const vtss_inst_t inst,
    vtss_fdma_list_t *const list )
```

Release DCBs.

This function returns DCBs (injection, extraction, or AFI) back to the FDMA driver.

This is useful in these situations: Extraction: If frame data memory is completely managed by the FDMA driver (`rx_alloc_cb() == NULL`), then the application may choose to pass the frame as is to higher levels of software, i.e. with zero-copy. Once it is handled, the DCBs must be returned to the FDMA driver with a call to this function. If frame data memory is managed by the application (`rx_alloc_cb() != NULL`), then the application should return the list of DCBs it was invoked with in the `rx_cb()` callback handler (with the `alloc_ptr` set to `NULL` if the frame itself is passed to higher levels of software).

Injection (both normal and AFI): This is rarely useful, and the only situation where it makes sense to return injection DCBs to the FDMA driver is when it has allocated the DCBs (using [vtss_fdma_dcb_get\(\)](#)) and then decides not to use them in a call to `vtss_fdma_inj()` afterwards.

- Call context:
Thread
- Re-entrant:
Yes

Parameters

<i>inst</i>	[IN]: Target instance reference.
<i>list</i>	[IN]: The list of DCBs to be returned to the application.

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR if a supplied parameter was erroneous.

7.11.5.4 vtss_fdma_tx()

```
vtss_rc vtss_fdma_tx (
    const vtss_inst_t inst,
    vtss_fdma_list_t * list,
```

```
vtss_fdma_tx_info_t *const fdma_info,
vtss_packet_tx_info_t *const tx_info )
```

Inject a frame.

This function takes a NULL-terminated list of software DCBs making up one frame and injects it using the tx info properties given by props.

The DCBs must be obtained using the [vtss_fdma_dcb_get\(\)](#) call.

Once the frame is injected, the configured tx_done_cb() function will be called. Once this function returns, the DCBs that the callback function was invoked with are no longer available to the application.

Upon invocation of tx_done_cb(), the application can read additional data from the frame's SOF DCB. The fields filled in by the FDMA driver are sw_tstamp, afi_frm_cnt, and afi_seq_number.

Upon tx_done_cb() the FDMA driver may have modified the actual frame data, and thereby the frm_ptr that was set during the call to [vtss_fdma_inj\(\)](#).

- Call context:
Thread
- Re-entrant:
Yes

Parameters

<i>inst</i>	[IN]: Target instance reference.
<i>list</i>	[IN]: The list of software DCBs making up the frame. Only the frm_ptr and act_len members must be filled in. The user member is useful for additional data required by the application to identify the frame that is being injected upon the tx_done_cb() callback.
<i>fdma_info</i>	[IN]: Info specifically for use by the FDMA driver. The structure may be allocated on the stack, since vtss_fdma_inj() doesn't keep a pointer to it after the call returns. Initialize with a call to vtss_fdma_tx_info_init() prior to filling it in.
<i>tx_info</i>	[IN]: Pointer to a structure that contains properties on how to inject the frame. Must be initialized with a call to vtss_packet_tx_info_init() before assigning it. The structure may be allocated on the stack, since vtss_fdma_inj() doesn't keep a pointer to it after the call returns.

Returns

VTSS_RC_OK on success.

VTSS_RC_ERROR on any kind of error. In this case, the DCBs are already returned to the FDMA driver.

7.11.5.5 [vtss_fdma_tx_info_init\(\)](#)

```
vtss_rc vtss_fdma_tx_info_init (
    const vtss_inst_t inst,
    vtss_fdma_tx_info_t *const fdma_info )
```

Initialize a [vtss_fdma_tx_info_t](#) structure.

Parameters

<i>inst</i>	[IN]: Target instance
<i>fdma_info</i>	[IN]: Structure to initialize

Returns

VTSS_RC_OK unless fdma_info == NULL.

7.11.5.6 vtss_fdma_afi_cancel()

```
vtss_rc vtss_fdma_afi_cancel (
    const vtss_inst_t inst,
    const u8 *const frm_ptr )
```

Cancel an ongoing AFI transmission.

The frame is identified by a pointer to the frame previously used in list->frm_ptr in the call to [vtss_fdma_tx\(\)](#).

Once the cancellation has occurred, the afi_tx_done callback will be called.

Parameters

<i>inst</i>	[IN]: Target instance
<i>frm_ptr</i>	[IN]: Frame pointer.

Returns

VTSS_RC_ERROR if no such frame was found. VTSS_RC_OK if cancelled successfully.

7.11.5.7 vtss_fdma_afi_frm_cnt()

```
vtss_rc vtss_fdma_afi_frm_cnt (
    const vtss_inst_t inst,
    const u8 *const frm_ptr,
    u64 *const frm_cnt )
```

Get an interim frame count for a given periodically injected frame.

The frame is identified by a pointer to the frame previously used in list->frm_ptr in the call to [vtss_fdma_tx\(\)](#). To get a count != 0, counting must be enabled with fdma_info.afi_enable_counting.

Parameters

<i>inst</i>	[IN]: Target instance
<i>frm_ptr</i>	[IN]: Frame pointer.
<i>frm_cnt</i>	[OUT]: Frame counter.

Returns

VTSS_RC_ERROR if no such frame was found. VTSS_RC_OK if successful.

7.11.5.8 vtss_fdma_dcb_get()

```
vtss_rc vtss_fdma_dcb_get (
    const vtss_inst_t inst,
    u32 dcb_cnt,
    vtss_fdma_dcb_type_t dcb_type,
    vtss_fdma_list_t ** list )
```

Get one or more DCBs suitable for frame injection.

The FDMA API is the owner/maintainer of all DCBs. In order to be able to inject a frame, the application must request DCBs from the FDMA driver and fill in appropriate fields (see description under `vtss_fdma_inj()`) prior to calling `vtss_fdma_inj()`.

One or more DCBs may be requested in one go. If the application regrets that it has requested the DCBs, it may call `vtss_fdma_dcbs_release()` to hand them back to the FDMA driver.

If more than one DCB is requested, the FDMA driver will hook them together with the DCBs' next field. The last DCB's next field will be set to NULL by the FDMA driver.

Special DCBs are required for AFI frames, so in addition to the number of DCBs, also a `dcb_type` parameter must be provided. The `dcb_type` must only be set to `VTSS_FDMA_DCB_TYPE_INJ` or `VTSS_FDMA_DCB_TYPE_AFI`.

If the requested number of DCBs are not available of the specified type, the function returns `VTSS_RC_INCOMPLETE`. It is up to the application to implement the logic that can facilitate waiting for DCBs, if it is required that a given frame must be transmitted. It could, e.g. wait for a semaphore in the thread that calls `vtss_fdma_inj()` and signal that semaphore whenever the `tx_done_cb()` gets invoked.

- Call context:
Thread
- Re-entrant:
Yes

Parameters

<i>inst</i>	[IN]: Target instance reference.
<i>dcb_cnt</i>	[IN]: Number of DCBs.
<i>dcb_type</i>	[IN]: DCB type requested.
<i>list</i>	[OUT]: Pointer receiving a pointer to the first DCB.

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR if a supplied parameter was erroneous.
 VTSS_RC_INCOMPLETE if `dcb_cnt` DCBs aren't available.

7.11.5.9 vtss_fdma_throttle_cfg_get()

```
vtss_rc vtss_fdma_throttle_cfg_get (
    const vtss_inst_t inst,
    vtss_fdma_throttle_cfg_t *const cfg )
```

Get current throttle configuration.

Returns the current throttling configuration.

- Call context:
Any
- Re-entrant:
Yes, but doesn't really make sense.

Parameters

<i>inst</i>	[IN]: Target instance reference.
<i>cfg</i>	[OUT]: Pointer to structure that receives the current throttling configuration.

Returns

VTSS_RC_OK on success, VTSS_RC_ERROR if channel indicated by ch is not configured for extraction.

7.11.5.10 vtss_fdma_throttle_cfg_set()

```
vtss_rc vtss_fdma_throttle_cfg_set (
    const vtss_inst_t inst,
    const vtss_fdma_throttle_cfg_t *const cfg )
```

Configure throttling.

Configure throttling. See [vtss_fdma_throttle_cfg_t](#) for a description of the throttle feature and the use of this function.

- Call context:
Any
- Re-entrant:
Yes, but doesn't really make sense.

Parameters

<i>inst</i>	[IN]: Target instance reference.
<i>cfg</i>	[IN]: New throttling configuration.

Returns

VTSS_RC_OK on success, VTSS_RC_ERROR if channel indicated by ch is not configured for extraction.

7.11.5.11 vtss_fdma_throttle_tick()

```
vtss_rc vtss_fdma_throttle_tick (
    const vtss_inst_t inst )
```

Generate throttle tick.

See [vtss_fdma_throttle_cfg_t](#) for a description of the throttle feature and the use of this function.

- Call context:
Any
- Re-entrant:
Yes, but doesn't really make sense to call this function from more than one thread.

Parameters

<i>inst</i>	[IN]: Target instance reference.
-------------	----------------------------------

Returns

VTSS_RC_OK - always.

7.11.5.12 vtss_fdma_stats_clr()

```
vtss_rc vtss_fdma_stats_clr (
    const vtss_inst_t inst )
```

Clear FDMA statistics.

- Call context:
Thread
- Re-entrant:
Yes

Parameters

<i>inst</i>	[IN]: Target instance reference.
-------------	----------------------------------

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR if supplied parameter was erroneous.

7.11.5.13 vtss_fdma_irq_handler()

```
vtss_rc vtss_fdma_irq_handler (
    const vtss_inst_t inst,
    void *const ctxt )
```

FDMA Interrupt Handler.

This function is the heart-beat of all FDMA silicon communication. The driver code enables interrupts on the FDMA silicon and expects this function to be called whenever the FDMA generates interrupts. The function is not re-entrant, and it must *not* be interrupted by other non-interrupt code. This function does not call any of the [VTSS_OS_INTERRUPT_DISABLE\(\)](#) or [VTSS_OS_SCHEDULER_LOCK\(\)](#) macros.

The function first checks for extraction channels interrupting and calls back the relevant xtr_cb() functions with one frame at a time. The function calls back for every frame that the FDMA has extracted since last call. Then it checks for injection completion, takes care of retransmission if that is needed, and calls back the relevant inj_post_cb() functions for every frame that has been injected.

This means that both xtr_cb() and inj_post_cb() callback functions are called from the same context as [vtss_fdma_irq_handler\(\)](#) is called. BEWARE!!

To better understand what is required by the caller of this function, let's branch on the two basic options:

- **Interrupt Driven Operation:**
 If interrupts are supported, it is important that the FDMA interrupt is disabled by the "top half" or ISR if it finds out that the interrupt is for the FDMA. If the OS supports top and bottom halves like Linux (a.k.a. I \leftrightarrow SRs and DSRs in eCos), it is recommended to call [vtss_fdma_irq_handler\(\)](#) in the bottom half/DSR rather than in the top half/ISR. In the bottom half/DSR case, it is ensured that no other bottom halves/DSRs or threads can preempt the handler. Once the handler returns, the caller should clear and re-enable top-level FDMA interrupts. In this scenario VTSS_OPT_FDMA_IRQ_CONTEXT should be defined to 0, causing the thread code (all other vtss_fdma_XXX() functions) to use the [VTSS_OS_SCHEDULER_LOCK\(\)](#)/[UNLOCK\(\)](#) functions.

If the OS doesn't support top/bottom halves, but still supports interrupt handling, [vtss_fdma_irq_handler\(\)](#) may be called directly from the ISR, which should still start by disabling top-level FDMA interrupts, then call the handler, and finally clear and re-enable them. In this scenario VTSS_OPT_FDMA_IRQ_CONTEXT be defined to 1. This causes the thread code (all other vtss_fdma_XXX() functions) to use the [VTSS_OS_INTERRUPT_DISABLE\(\)](#)/[RESTORE\(\)](#) functions whenever it is using or updating state members also used by the interrupt handler.

- **Polled Operation:**
 Since the FDMA driver code reads registers directly in the FDMA silicon to figure out which channels are interrupting, and since it allows for "no channels want the CPU's attention", it is possible to use a polled approach rather than an interrupt driven approach. In a single-threaded application you can call it once in the round-robin trip, because it cannot be interrupted by any other code. In a multi-threaded application you will have to disable the scheduler before calling it, and re-enable it afterwards. This ensures that no other functions can call e.g. vtss_fdma_inj() while servicing the "interrupt". It is not good enough to disable interrupts, because the code calls macros like "output this debug message to the console", which may end up in the OS kernel, which in turn may schedule the [vtss_fdma_irq_handler\(\)](#) function out in favor of another thread.
- **Call context:**
 OS-Specific. See description below.
- **Re-entrant:**
NO!!!

Parameters

<i>inst</i>	[IN]: Target instance reference.
<i>cntxt</i>	[IN]: A user-defined argument, which is passed to the inj_post_cb() and xtr_cb() callback functions. Rarely needed.

Returns

VTSS_RC_OK on success.
 VTSS_RC_ERROR if @inst parameter was erroneous.

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`
`= 0x17424, VTSS_TARGET_CARACAL_1 = 0x7428,`
`VTSS_TARGET_CARACAL_2 = 0x7429, VTSS_TARGET_JAGUAR_1 = 0x7460, VTSS_TARGET_LYNX_1`
`= 0x7462, VTSS_TARGET_E_STAX_III_48 = 0x7432,`
`VTSS_TARGET_E_STAX_III_68 = 0x7434, VTSS_TARGET_E_STAX_III_24_DUAL = 0xD7431,`
`VTSS_TARGET_E_STAX_III_68_DUAL = 0xD7434, VTSS_TARGET_DAYTONA = 0x8492,`
`VTSS_TARGET_TALLADEGA = 0x8494, VTSS_TARGET_SERVAL_2 = 0x7438, VTSS_TARGET_LYNX_2`
`= 0x7464, VTSS_TARGET_JAGUAR_2 = 0x7468,`
`VTSS_TARGET_SPARX_IV_52 = 0x7442, VTSS_TARGET_SPARX_IV_44 = 0x7444, VTSS_TARGET_SPARX_IV_80`
`= 0x7448, VTSS_TARGET_SPARX_IV_90 = 0x7449 }`
Target chip type.
- `enum vtss_pi_width_t { VTSS_PI_WIDTH_16 = 0, VTSS_PI_WIDTH_8 }`
PI data width.
- `enum vtss_port_mux_mode_t { VTSS_PORT_MUX_MODE_0, VTSS_PORT_MUX_MODE_1, VTSS_PORT_MUX_MODE_2 }`
Port mux configuration.
- `enum vtss_restart_info_src_t { VTSS_RESTART_INFO_SRC_NONE, VTSS_RESTART_INFO_SRC_CU ↔ _PHY, VTSS_RESTART_INFO_SRC_10G_PHY }`
Restart information source.
- `enum vtss_restart_t { VTSS_RESTART_COLD, VTSS_RESTART_COOL, VTSS_RESTART_WARM }`
Restart type.

Functions

- `vtss_rc vtss_inst_get (const vtss_target_type_t target, vtss_inst_create_t *const create)`
Initialize create structure for target.
- `vtss_rc vtss_inst_create (const vtss_inst_create_t *const create, vtss_inst_t *const inst)`
Create target instance.
- `vtss_rc vtss_inst_destroy (const vtss_inst_t inst)`
Destroy target instance.
- `vtss_rc vtss_init_conf_get (const vtss_inst_t inst, vtss_init_conf_t *const conf)`
Get default initialization configuration.
- `vtss_rc vtss_init_conf_set (const vtss_inst_t inst, const vtss_init_conf_t *const conf)`
Set initialization configuration.
- `vtss_rc vtss_restart_conf_end (const vtss_inst_t inst)`
Indicate configuration end. If a warm start has been done, the stored configuration will be applied.
- `vtss_rc vtss_restart_status_get (const vtss_inst_t inst, vtss_restart_status_t *const status)`
Get restart status.
- `vtss_rc vtss_restart_conf_get (const vtss_inst_t inst, vtss_restart_t *const restart)`
Get restart configuration (next restart mode)
- `vtss_rc vtss_restart_conf_set (const vtss_inst_t inst, const vtss_restart_t restart)`
Set restart configuration (next restart mode)

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_TYPE_ETYPE`, `VTSS_VCE_TYPE_LLC`, `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] )
```

Set 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 vtss_vlan_trans_grp2vlan_conf_t . Input group_id in the conf structure
<i>next</i>	[IN] Flag to indicate next entry.

Returns

Return code.

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 vtss_vlan_trans_port2grp_conf_t .

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 vtss_vlan_trans_port2grp_conf_t .
<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 relates 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_sgpio_mode_t` {

 `VTSS_SGPIO_MODE_OFF`, `VTSS_SGPIO_MODE_ON`, `VTSS_SGPIO_MODE_0`, `VTSS_SGPIO_MODE_1`,

 `VTSS_SGPIO_MODE_0_ACTIVITY`, `VTSS_SGPIO_MODE_1_ACTIVITY`, `VTSS_SGPIO_MODE_0_ACTIVITY_INV`,

 `VTSS_SGPIO_MODE_1_ACTIVITY_INV` }

SGPIO output mode.
- enum `vtss_sgpio_bmode_t` {

 `VTSS_SGPIO_BMODE_TOGGLE`, `VTSS_SGPIO_BMODE_0_625`, `VTSS_SGPIO_BMODE_1_25`,

 `VTSS_SGPIO_BMODE_2_5`,

 `VTSS_SGPIO_BMODE_5` }

SGPIO 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_sgpio_conf_get` (const `vtss_inst_t` inst, const `vtss_chip_no_t` chip_no, const `vtss_sgpio_group_t` group, `vtss_sgpio_conf_t` *const conf)

Get SGPIO configuration.
- `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 SGPIO configuration.
- `vtss_rc vtss_sgpio_read` (const `vtss_inst_t` inst, const `vtss_chip_no_t` chip_no, const `vtss_sgpio_group_t` group, `vtss_sgpio_port_data_t` data[`VTSS_SGPIO_PORTS`])

Read SGPIO data.
- `vtss_rc vtss_sgpio_event_poll` (const `vtss_inst_t` inst, const `vtss_chip_no_t` chip_no, const `vtss_sgpio_group_t` group, const `u32` bit, `BOOL` *const events)

Get SGPIO event indication.
- `vtss_rc vtss_sgpio_event_enable` (const `vtss_inst_t` inst, const `vtss_chip_no_t` chip_no, const `vtss_sgpio_group_t` group, const `vtss_port_no_t` port, const `u32` bit, `BOOL` enable)

Get SGPIO event enable.
- `vtss_rc vtss_intr_cfg` (const `vtss_inst_t` inst, const `u32` mask, const `BOOL` polarity, const `BOOL` enable)

Configure interrupt.
- `vtss_rc vtss_intr_mask_set` (const `vtss_inst_t` inst, `vtss_intr_t` *mask)

Set the interrupt mask.
- `vtss_rc vtss_intr_status_get` (const `vtss_inst_t` inst, `vtss_intr_t` *status)

Get the interrupt status for all enabled sources. The interrupt status bit is cleared by the function.
- `vtss_rc vtss_intr_pol_negation` (const `vtss_inst_t` inst)

This will negate polarity on fast link fail detection signals when active. This is only intended to be used on Luton26 RevB as a work around for the Atom PHY interrupt always active problem.
- `vtss_rc vtss_irq_conf_get` (const `vtss_inst_t` inst, const `vtss_irq_t` irq, `vtss_irq_conf_t` *conf)

Get IRQ configuration.
- `vtss_rc vtss_irq_conf_set` (const `vtss_inst_t` inst, const `vtss_irq_t` irq, const `vtss_irq_conf_t` *const conf)

Set IRQ configuration.
- `vtss_rc vtss_irq_status_get_and_mask` (const `vtss_inst_t` inst, `vtss_irq_status_t` *status)

Get IRQ status (active sources), mask current sources.
- `vtss_rc vtss_irq_enable` (const `vtss_inst_t` inst, const `vtss_irq_t` irq, `BOOL` enable)

Control a specific interrupt source.
- `u32 vtss_tod_get_ns_cnt` (void)

Get the current hw nanosec time. This function is called from interrupt.
- `void vtss_tod_set_ns_cnt_cb` (`tod_get_ns_cnt_cb_t` cb)

Set an external hw nanosec read function.
- `vtss_rc vtss_temp_sensor_init` (const `vtss_inst_t` inst, const `BOOL` enable)

Initialize the temperature sensor.
- `vtss_rc vtss_temp_sensor_get` (const `vtss_inst_t` inst, `i16` *temperature)

Read temperature sensor value.
- `vtss_rc vtss_fan_rotation_get` (const `vtss_inst_t` inst, `vtss_fan_conf_t` *const fan_spec, `u32` *rotation_count)

Get the number of fan rotations.
- `vtss_rc vtss_fan_cool_lvl_set` (const `vtss_inst_t` inst, `u8` lvl)

Set fan cool level (Duty cycle)

- `vtss_rc vtss_fan_controller_init` (const `vtss_inst_t` inst, const `vtss_fan_conf_t` *const spec)
Initialise fan controller
- `vtss_rc vtss_fan_cool_lvl_get` (const `vtss_inst_t` inst, `u8` *lvl)
Get fan cool level (Duty cycle)
- `vtss_rc vtss_eee_port_conf_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_eee_port_conf_t` *const eee_conf)
Set EEE configuration.
- `vtss_rc vtss_eee_port_state_set` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, const `vtss_eee_port_state_t` *const eee_state)
Change EEE Port state.
- `vtss_rc vtss_eee_port_counter_get` (const `vtss_inst_t` inst, const `vtss_port_no_t` port_no, `vtss_eee_port_counter_t` *const eee_counter)
Get EEE-related port counters.
- `vtss_rc vtss_debug_reg_check_set` (const `vtss_inst_t` inst, const `BOOL` enable)
Enable or disable register access checking.

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

VTSS_SGPIO_BMODE_TOGGLE	Burst toggle (mode 1 only)
VTSS_SGPIO_BMODE_0_625	0.625 Hz (mode 0 only)
VTSS_SGPIO_BMODE_1_25	1.25 Hz
VTSS_SGPIO_BMODE_2_5	2.5 Hz
VTSS_SGPIO_BMODE_5	5 Hz

Definition at line 779 of file vtss_misc_api.h.

7.19.4.9 vtss_irq_t

enum `vtss_irq_t`

Interrupt sources.

Enumerator

VTSS_IRQ_XTR	Frame Extraction Ready(register-based)
VTSS_IRQ_FDMA_XTR	Frame Extraction Ready (FDMA-based)
VTSS_IRQ_SOFTWARE	Software IRQ
VTSS_IRQ_PTP_RDY	PTP Timestamp Ready
VTSS_IRQ_PTP_SYNC	PTP Synchronization IRQ
VTSS_IRQ_EXT1	EXT1 IRQ
VTSS_IRQ_OAM	OAM IRQ
VTSS_IRQ_MAX	Maximum IRQ Source

Definition at line 957 of file vtss_misc_api.h.

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_ecos.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.

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.

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 vtss_mem_flags_t 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 vtss_callout_malloc() .
<i>flags</i>	[IN] See vtss_mem_flags_t 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.

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_TWOSTEP	Both one- and two-step PTP
VTSS_PACKET_PTP_ACTION_ORIGIN_TIMESTAMPAMP	Update time-of-day in PTP frame's originTimestamp field

Definition at line 543 of file vtss_packet_api.h.

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>Can be set by:</p> <p>Luton26: Y Jaguar1: Y Serval: Y Jaguar2: Y Serval2: Y ServalT: Y</p>
VTSS_PACKET_RX_HINTS_VLAN_FRAME_MISMATCH	<p>If a tagged frame is received on a port that only should accept untagged frames or if an untagged frame is received on a port that should only accept tagged frames, then this will be set. In general, the application should not drop such frames, because e.g. BPDUs are normally untagged.</p> <p>Can be set by:</p> <p>Luton26: Y Jaguar1: Y Serval: Y Jaguar2: Y Serval2: Y ServalT: Y</p>
VTSS_PACKET_RX_HINTS_VID_MISMATCH	<p>If a frame gets classified to a VLAN ID on a port that is not member of that VID, the hints flags include this enum.</p> <p>Can be set by:</p> <p>Luton26: Y Jaguar1: Y Serval: Y Jaguar2: Y Serval2: Y ServalT: Y</p>

Definition at line 915 of file vtss_packet_api.h.

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_ci37_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)

- `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 100m.
- `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)`

Function to get the Mean Squared Error (MSE), ie. Noise floor for 1000m.
- `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.

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 pArtners's ability.

Parameters

<i>inst</i>	[IN] Target instance reference.
<i>port_no</i>	[IN] Port number.
<i>status</i>	[OUT] PHY status.

Returns

Return code.

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 vtss_phy_wol_conf_t 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 vtss_phy_wol_conf_t 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 <i>mcb_bus</i> is returning 0 if the data is for 1G, and returning 1 if the data is for 6G .
<i>cfg_buf</i>	[IN] Pointer to array where to put the current settings.
<i>stat_buf</i>	[IN] Pointer to array where to put the current status.

Returns

Return code. VTSS_RC_OK if patch were updated correct else VTSS_RC_ERROR

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 vtss_rcpll_status_t 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 vtss_rcpll_status_t 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 vtss_lcpll_status_t 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 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.

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 vtss_sync_clock_out_t 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 vtss_sync_clock_out_t 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 vtss_sync_clock_in_t 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 vtss_sync_clock_in_t 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 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, 217

acl_hit
 vtss_packet_rx_info_t, 155

acl_idx
 vtss_packet_rx_info_t, 155

act_len
 tag_vtss_fdma_list, 28

action
 vtss_ace_t, 50
 vtss_acl_port_conf_t, 58
 vtss_ece_t, 81
 vtss_mce_t, 138
 vtss_qce_t, 262
 vtss_vce_t, 321

active
 vtss_irq_status_t, 124

addr
 vtss_ip_addr_t, 117
 vtss_ipv6_t, 122
 vtss_mac_t, 131
 vtss_wol_mac_addr_t, 332

address
 vtss_ip_network_t, 118
 vtss_ipv4_network_t, 119
 vtss_ipv6_network_t, 121

adv_dis
 port_custom_conf_t, 25

advertisement
 vtss_port_clause_37_control_t, 219

afi_buf_cnt
 vtss_fdma_cfg_t, 100

afi_done_cb
 vtss_fdma_cfg_t, 101

afi_enable_counting
 vtss_fdma_tx_info_t, 109

afi_enable_sequence_numbering
 vtss_fdma_tx_info_t, 109

afi_fps
 vtss_fdma_tx_info_t, 108

afi_frm_cnt
 tag_vtss_fdma_list, 30

afi_seq_number
 tag_vtss_fdma_list, 30

afi_sequence_number_offset
 vtss_fdma_tx_info_t, 110

afi_type
 vtss_fdma_tx_info_t, 109

aged

vtss_mac_table_entry_t, 132
vtss_mac_table_status_t, 134

aggr_code
 vtss_packet_tx_info_t, 174

aggr_rx_disable
 vtss_packet_frame_info_t, 144
 vtss_packet_port_info_t, 146

aggr_tx_disable
 vtss_packet_frame_info_t, 144
 vtss_packet_port_info_t, 146

alloc_ptr
 tag_vtss_fdma_list, 29

aneg
 vtss_phy_conf_t, 186
 vtss_port_status_t, 248

aneg_complete
 vtss_port_sgmii_aneg_t, 246
 vtss_port_status_t, 247

aneg_enable
 vtss_phy_tbi_conf_t, 207

aneg_pd_detect
 vtss_phy_media_serdes_pcs_ctrl_t, 198

aneg_restart
 vtss_phy_mac_serdes_pcs_ctrl_t, 195

arp
 vtss_ace_frame_arp_t, 31
 vtss_ace_t, 51

arrived_tagged
 vtss_packet_rx_header_t, 149

asymmetric_pause
 vtss_phy_aneg_t, 182
 vtss_port_clause_37_adv_t, 217

automatic
 vtss_learn_mode_t, 130

autoneg
 port_custom_conf_t, 24

BOOL
 types.h, 393

base_port_no
 vtss_phy_type_t, 209

bit_count
 vtss_sgpio_conf_t, 282

bit_rate
 vtss_acl_policer_conf_t, 57

bit_rate_enable
 vtss_acl_policer_conf_t, 57

bmode
 vtss_sgpio_conf_t, 282

bpdu_cpu_only

vtss_packet_rx_reg_t, 169
bpd़u_queue
 vtss_packet_rx_queue_map_t, 166
bpd़u_reg
 vtss_packet_rx_port_conf_t, 164
bridge
 vtss_port_counters_t, 224
byte_limit_per_tick
 vtss_fdma_throttle_cfg_t, 106
bytes
 vtss_counter_pair_t, 65
CHIP_PORT_UNUSED
 vtss_port_api.h, 664
cal_done
 vtss_lcpll_status_t, 128
cal_error
 vtss_lcpll_status_t, 128
 vtss_rcpll_status_t, 275
cal_not_done
 vtss_rcpll_status_t, 275
cb
 vtss_ts_timestamp_alloc_t, 296
cbs
 vtss_dlb_policer_conf_t, 70
ccm_quotient_max
 vtss_fdma_ch_cfg_t, 105
cf
 vtss_dlb_policer_conf_t, 69
cfg
 vtss_phy_conf_1g_t, 184
cfi
 vtss_ace_vlan_t, 53
 vtss_tci_t, 288
channel_id
 vtss_phy_type_t, 209
chip_no
 vtss_debug_info_t, 66
 vtss_debug_lock_t, 68
 vtss_fdma_ch_cfg_t, 104
 vtss_packet_rx_meta_t, 160
 vtss_port_map_t, 235
chip_port
 vtss_port_map_t, 234
cir
 vtss_dlb_policer_conf_t, 70
clear
 vtss_debug_info_t, 67
connector_enable
 vtss_phy_loopback_t, 193
context
 vtss_ts_timestamp_alloc_t, 296
 vtss_ts_timestamp_t, 297
copper
 vtss_port_status_t, 248
copy_to_cpu
 vtss_mac_table_entry_t, 132
cos
 vtss_packet_rx_info_t, 154
cpu
 vtss_acl_action_t, 54
 vtss_learn_mode_t, 130
cpu_once
 vtss_acl_action_t, 54
cpu_queue
 vtss_acl_action_t, 54
 vtss_mac_table_entry_t, 132
cs_wait_ns
 vtss_pi_conf_t, 213
cu_bad
 vtss_phy_statistic_t, 205
cu_good
 vtss_phy_statistic_t, 204
cur_version
 vtss_restart_status_t, 276
data
 vtss_ace_frame_etype_t, 34
 vtss_ace_frame_ipv4_t, 37
 vtss_ace_frame_ipv6_t, 41
 vtss_mce_key_t, 137
 vtss_qce_frame_etype_t, 252
 vtss_qce_frame_llc_t, 256
 vtss_qce_frame_snap_t, 257
 vtss_vce_frame_etype_t, 312
 vtss_vce_frame_llc_t, 316
 vtss_vce_frame_snap_t, 316
default_dei
 vtss_qos_port_conf_t, 271
default_dpl
 vtss_qos_port_conf_t, 271
default_prio
 vtss_qos_port_conf_t, 270
dei
 vtss_ece_outer_tag_t, 80
 vtss_ece_tag_t, 83
 vtss_evc_inner_tag_t, 91
 vtss_qce_action_t, 251
 vtss_qce_tag_t, 263
 vtss_vce_tag_t, 322
 vtss_vlan_tag_t, 328
dei_colouring
 vtss_evc_port_conf_t, 93
destination
 vtss_ipv4_uc_t, 120
 vtss_ipv6_uc_t, 122
 vtss_irq_conf_t, 123
 vtss_mac_table_entry_t, 132
dgroup_no
 vtss_dgroup_port_conf_t, 68
dip
 vtss_ace_frame_arp_t, 33
 vtss_ace_frame_ipv4_t, 37
dir
 vtss_ece_action_t, 71
disable
 vtss_phy_mac_serd_pcns_ctrl_t, 195

discard
 vtss_learn_mode_t, 130

divider
 vtss_sync_clock_out_t, 287

dma_enable
 vtss_packet_dma_conf_t, 142

dmac
 vtss_ace_frame_etype_t, 34
 vtss_ace_frame_llc_t, 45
 vtss_ace_frame_snap_t, 46

dmac_bc
 vtss_ace_t, 50
 vtss_ece_mac_t, 79
 vtss_qce_mac_t, 260
 vtss_vce_mac_t, 320

dmac_dip
 vtss_evc_port_conf_t, 94
 vtss_qos_port_conf_t, 274
 vtss_vcl_port_conf_t, 323

dmac_enable
 vtss_aggr_mode_t, 59

dmac_match
 vtss_ace_frame_arp_t, 32

dmac_mc
 vtss_ace_t, 50
 vtss_ece_mac_t, 79
 vtss_qce_mac_t, 260
 vtss_vce_mac_t, 320

dot1dTpPortInDiscards
 vtss_port_bridge_counters_t, 216

dot3InPauseFrames
 vtss_port_ethernet_like_counters_t, 225

dot3OutPauseFrames
 vtss_port_ethernet_like_counters_t, 226

dp
 vtss_packet_tx_info_t, 178
 vtss_qce_action_t, 250

dp_enable
 vtss_qce_action_t, 250

dp_level_map
 vtss_qos_port_conf_t, 272

dport
 vtss_ace_frame_ipv4_t, 37
 vtss_ace_frame_ipv6_t, 42
 vtss_ece_frame_ipv4_t, 74
 vtss_ece_frame_ipv6_t, 76
 vtss_qce_frame_ipv4_t, 254
 vtss_qce_frame_ipv6_t, 255
 vtss_vce_frame_ipv4_t, 313
 vtss_vce_frame_ipv6_t, 315

ds
 vtss_ace_frame_ipv4_t, 36
 vtss_ace_frame_ipv6_t, 41

dscp
 vtss_ece_frame_ipv4_t, 73
 vtss_ece_frame_ipv6_t, 75
 vtss_qce_action_t, 250
 vtss_qce_frame_ipv4_t, 253

 vtss_qce_frame_ipv6_t, 255
 vtss_vce_frame_ipv4_t, 313
 vtss_vce_frame_ipv6_t, 314

 vtss_vce_frame_outer_tag_t, 80
 vtss_fdma_cfg_t, 96
 vtss_npi_conf_t, 141
 vtss_packet_rx_queue_npi_conf_t, 168

 vtss_qce_action_t, 250

 vtss_qos_port_conf_t, 272

 vtss_qos_class_map
 vtss_qos_conf_t, 265

 vtss_qos_map
 vtss_qos_conf_t, 265

 vtss_qos_map_dp1
 vtss_qos_conf_t, 265

 vtss_qos_remap
 vtss_qos_conf_t, 266

 vtss_qos_remap_dp1
 vtss_qos_conf_t, 266

 vtss_qos_remark
 vtss_qos_conf_t, 266

 vtss_qos_translate
 vtss_qos_port_conf_t, 272

 vtss_qos_translate_map
 vtss_qos_conf_t, 266

 vtss_qos_trust
 vtss_qos_conf_t, 265

dst_port_mask
 vtss_packet_tx_info_t, 172

dual_media_fiber_speed
 port_custom_conf_t, 25

dwrr_enable
 vtss_qos_port_conf_t, 274

ebs
 vtss_dlb_policer_conf_t, 70

eee_ena
 vtss_eee_port_conf_t, 84

eee_ena_phy
 vtss_phy_eee_conf_t, 188

eee_fast_queues
 vtss_eee_port_conf_t, 84

eee_mode
 vtss_phy_eee_conf_t, 188

eir
 vtss_dlb_policer_conf_t, 70

enable
 port_custom_conf_t, 23
 vtss_ace_ptp_t, 47
 vtss_ace_sip_smac_t, 48
 vtss_dlb_policer_conf_t, 69
 vtss_ece_outer_tag_t, 80
 vtss_fdma_cfg_t, 96
 vtss_npi_conf_t, 141
 vtss_packet_rx_queue_npi_conf_t, 168

vtss_port_clause_37_control_t, 218
 vtss_sync_clock_in_t, 286
 vtss_sync_clock_out_t, 287
 vtss_ts_ext_clock_mode_t, 292
 enabled
 vtss_sgpi_port_conf_t, 283
 ethernet
 vtss_ace_frame_arp_t, 33
 ethernet_like
 vtss_port_counters_t, 224
 etype
 vtss_ace_frame_etype_t, 34
 vtss_ace_t, 51
 vtss_packet_rx_meta_t, 161
 vtss_qce_frame_etype_t, 252
 vtss_qce_key_t, 258
 vtss_vce_frame_etype_t, 311
 vtss_vce_key_t, 318
 evc
 vtss_port_counters_t, 225
 evc_id
 vtss_ece_action_t, 72
 evc_police
 vtss_acl_action_t, 55
 evc_policer_id
 vtss_acl_action_t, 55
 exc_col_cont
 port_custom_conf_t, 25
 vtss_port_conf_t, 222
 excess_enable
 vtss_qos_port_conf_t, 270
 external
 vtss_irq_conf_t, 123
 FALSE
 types.h, 375
 fan_low_pol
 vtss_fan_conf_t, 95
 fan_open_col
 vtss_fan_conf_t, 95
 fan_pwm_freq
 vtss_fan_conf_t, 95
 far_end_enable
 vtss_phy_loopback_t, 192
 fast_link_stat_ena
 vtss_phy_mac_serdes_pcs_ctrl_t, 197
 fcs
 vtss_packet_rx_meta_t, 161
 fdx
 port_custom_conf_t, 24
 vtss_phy_forced_t, 190
 vtss_port_clause_37_adv_t, 217
 vtss_port_conf_t, 221
 vtss_port_sgmii_aneg_t, 245
 vtss_port_status_t, 247
 fdx_gap
 vtss_port_frame_gaps_t, 230
 fiber
 vtss_port_status_t, 248
 file
 vtss_api_lock_t, 62
 fill_level
 vtss_eee_port_counter_t, 86
 fill_level_get
 vtss_eee_port_counter_t, 85
 fill_level_thres
 vtss_eee_port_counter_t, 86
 filter
 vtss_packet_port_filter_t, 145
 flf
 vtss_phy_conf_t, 186
 flow_control
 port_custom_conf_t, 24
 vtss_policer_ext_t, 214
 vtss_port_conf_t, 221
 force
 vtss_phy_reset_conf_t, 202
 force_adv_ability
 vtss_phy_mac_serdes_pcs_ctrl_t, 195
 vtss_phy_media_serdes_pcs_ctrl_t, 198
 force_ams_sel
 vtss_phy_conf_t, 187
 force_fefi
 vtss_phy_media_serdes_pcs_ctrl_t, 199
 force_fefi_value
 vtss_phy_media_serdes_pcs_ctrl_t, 199
 force_hls
 vtss_phy_media_serdes_pcs_ctrl_t, 199
 forced
 vtss_phy_conf_t, 185
 fragment
 vtss_ace_frame_ipv4_t, 36
 vtss_ece_frame_ipv4_t, 73
 vtss_qce_frame_ipv4_t, 253
 vtss_vce_frame_ipv4_t, 312
 frame
 vtss_ace_t, 52
 vtss_ece_key_t, 78
 vtss_qce_key_t, 259
 vtss_vce_key_t, 319
 frame_gaps
 vtss_port_conf_t, 220
 frame_length_chk
 port_custom_conf_t, 26
 vtss_port_conf_t, 222
 frame_rate
 vtss_policer_ext_t, 214
 frame_type
 vtss_vlan_port_conf_t, 327
 frames
 vtss_counter_pair_t, 65
 freq
 vtss_phy_clock_conf_t, 183
 vtss_ts_ext_clock_mode_t, 293
 frm_len
 vtss_packet_tx_info_t, 173
 frm_limit_per_tick

vtss_fdma_throttle_cfg_t, 106
frm_ptr
tag_vtss_fdma_list, 28
fsm_lock
vtss_lcpoll_status_t, 128
fsm_stat
vtss_lcpoll_status_t, 129
full
vtss_debug_info_t, 66
function
vtss_api_lock_t, 62
fwd_enable
vtss_mirror_conf_t, 139

gain_stat
vtss_lcpoll_status_t, 129
garp_cpu_only
vtss_packet_rx_reg_t, 169
garp_queue
vtss_packet_rx_queue_map_t, 166
garp_reg
vtss_packet_rx_port_conf_t, 164
generate
vtss_port_flow_control_conf_t, 229
generate_pause
vtss_aneg_t, 61
glag_no
vtss_packet_rx_info_t, 152
group
vtss_debug_info_t, 66
group_id
vtss_vlan_trans_grp2vlan_conf_t, 329
vtss_vlan_trans_port2grp_conf_t, 330
grp_map
vtss_packet_rx_conf_t, 148

hdx
vtss_port_clause_37_adv_t, 217
vtss_port_sgmii_aneg_t, 245
hdx_gap_1
vtss_port_frame_gaps_t, 230
hdx_gap_2
vtss_port_frame_gaps_t, 230
header
vtss_ace_ptp_t, 47
high
vtss_vcap_udp_tcp_t, 307
vtss_vcap_vr_t, 309
hints
vtss_packet_rx_info_t, 151
hw_cnt
vtss_os_timestamp_t, 142
hw_tstamp
vtss_packet_rx_info_t, 156
hw_tstamp_decoded
vtss_packet_rx_info_t, 157

i16
types.h, 392

i32
types.h, 392
i64
types.h, 392
i8
types.h, 392
i_cpu_en
vtss_phy_reset_conf_t, 203
ib_cterm_ena
vtss_serdes_macro_conf_t, 280
id
vtss_ace_t, 49
vtss_ece_t, 81
vtss_mce_t, 138
vtss_qce_t, 261
vtss_ts_timestamp_t, 297
vtss_vce_t, 321
if_group
vtss_port_counters_t, 224
if_type
vtss_port_conf_t, 220
ifInBroadcastPkts
vtss_port_if_group_counters_t, 232
ifInDiscards
vtss_port_if_group_counters_t, 232
ifInErrors
vtss_port_if_group_counters_t, 232
ifInMulticastPkts
vtss_port_if_group_counters_t, 231
ifInNUcastPkts
vtss_port_if_group_counters_t, 232
ifInOctets
vtss_port_if_group_counters_t, 231
ifInUcastPkts
vtss_port_if_group_counters_t, 231
ifOutBroadcastPkts
vtss_port_if_group_counters_t, 233
ifOutDiscards
vtss_port_if_group_counters_t, 233
ifOutErrors
vtss_port_if_group_counters_t, 234
ifOutMulticastPkts
vtss_port_if_group_counters_t, 233
ifOutNUcastPkts
vtss_port_if_group_counters_t, 233
ifOutOctets
vtss_port_if_group_counters_t, 232
ifOutUcastPkts
vtss_port_if_group_counters_t, 233
ifh
vtss_packet_tx_ifh_t, 170
igmp_cpu_only
vtss_packet_rx_reg_t, 169
igmp_queue
vtss_packet_rx_queue_map_t, 166
in_range
vtss_vcap_udp_tcp_t, 306
ingress_filter

vtss_vlan_port_conf_t, 327
 inhibit_odd_start
 vtss_phy_mac_serdes_pcs_ctrl_t, 197
 vtss_phy_media_serdes_pcs_ctrl_t, 199
 inj_grp_mask
 vtss_fdma_ch_cfg_t, 102
 inner_tag
 vtss_evc_pb_conf_t, 93
 vtss_evc_port_conf_t, 94
 inst
 vtss_api_lock_t, 62
 int_fmt
 vtss_ts_internal_mode_t, 294
 int_pol_high
 vtss_sgpioport_conf_t, 283
 ip
 vtss_ace_frame_arp_t, 32
 ipmc_ctrl_cpu_copy
 vtss_packet_rx_reg_t, 169
 ipmc_ctrl_queue
 vtss_packet_rx_queue_map_t, 166
 ipv4
 vtss_ace_t, 51
 vtss_ece_key_t, 77
 vtss_ip_addr_t, 117
 vtss_qce_key_t, 259
 vtss_vce_key_t, 318
 ipv4_uc
 vtss_routing_entry_t, 277
 ipv4uc_received_frames
 vtss_l3_counters_t, 126
 ipv4uc_received_octets
 vtss_l3_counters_t, 126
 ipv4uc_transmitted_frames
 vtss_l3_counters_t, 127
 ipv4uc_transmitted_octets
 vtss_l3_counters_t, 126
 ipv6
 vtss_ace_t, 52
 vtss_ece_key_t, 78
 vtss_ip_addr_t, 117
 vtss_qce_key_t, 259
 vtss_vce_key_t, 319
 ipv6_uc
 vtss_routing_entry_t, 278
 ipv6uc_received_frames
 vtss_l3_counters_t, 126
 ipv6uc_received_octets
 vtss_l3_counters_t, 126
 ipv6uc_transmitted_frames
 vtss_l3_counters_t, 127
 ipv6uc_transmitted_octets
 vtss_l3_counters_t, 127
 isidx
 vtss_packet_rx_info_t, 158
 vtss_packet_tx_info_t, 177
 isdx_dont_use
 vtss_packet_tx_info_t, 178
 ivid
 vtss_evc_pb_conf_t, 92
 key
 vtss_ece_t, 81
 vtss_mce_t, 138
 vtss_qce_t, 262
 vtss_vce_t, 321
 l2_types.h
 vtss_sflow_type_t, 333
 latch_timestamp
 vtss_packet_tx_info_t, 176
 layer
 vtss_debug_info_t, 66
 lb_type
 vtss_phy_api.h, 614
 learn
 vtss_acl_action_t, 55
 vtss_packet_rx_header_t, 149
 learn_queue
 vtss_packet_rx_queue_map_t, 166
 learned
 vtss_mac_table_status_t, 133
 learning
 vtss_evc_conf_t, 89
 vtss_vlan_vid_conf_t, 331
 length
 vtss_ace_frame_arp_t, 32
 vtss_packet_rx_header_t, 148
 vtss_packet_rx_info_t, 151
 vtss_packet_rx_meta_t, 162
 vtss_packet_tx_ifh_t, 170
 vtss_phy_veriphy_result_t, 210
 level
 vtss_phy_power_status_t, 201
 vtss_policer_t, 215
 vtss_shaper_t, 285
 vtss_trace_conf_t, 292
 line
 vtss_api_lock_t, 62
 line_rate
 vtss_dlb_policer_conf_t, 70
 link
 vtss_phy_veriphy_result_t, 210
 vtss_port_sgmii_aneg_t, 245
 vtss_port_status_t, 247
 link_change
 vtss_intr_t, 116
 link_down
 vtss_port_status_t, 247
 list
 vtss_fdma_ch_cfg_t, 103
 llabs
 vtss_os_ecos.h, 553
 llc
 vtss_ace_frame_llc_t, 45
 vtss_ace_t, 51
 vtss_qce_key_t, 259

vtss_vce_key_t, 318
lock_status
 vtss_lcpoll_status_t, 128
locked
 vtss_mac_table_entry_t, 132
loop
 vtss_port_conf_t, 223
low
 vtss_vcap_udp_tcp_t, 306
 vtss_vcap_vr_t, 309
lp_advertisement
 vtss_eee_port_conf_t, 84
lrn_all_queue
 vtss_packet_rx_queue_map_t, 167

MAC_ADDR_BROADCAST
 types.h, 383
MAX_CFG_BUF_SIZE
 vtss_phy_api.h, 594
MAX_STAT_BUF_SIZE
 vtss_phy_api.h, 594
MAX_WOL_MAC_ADDR_SIZE
 vtss_phy_api.h, 605
MAX_WOL_PASSWD_SIZE
 vtss_phy_api.h, 605
MIN_WOL_PASSWD_SIZE
 vtss_phy_api.h, 605
mac
 vtss_ece_key_t, 77
 vtss_qce_key_t, 258
 vtss_vce_key_t, 317
 vtss_vid_mac_t, 324
mac_if
 vtss_phy_reset_conf_t, 202
mac_if_pcs
 vtss_phy_conf_t, 187
mac_serdes_equipment_enable
 vtss_phy_loopback_t, 193
mac_serdes_facility_enable
 vtss_phy_loopback_t, 193
mac_serdes_input_enable
 vtss_phy_loopback_t, 193
mac_vid_queue
 vtss_packet_rx_queue_map_t, 166
magic_pkt_cnt
 vtss_phy_wol_conf_t, 212
map
 vtss_packet_rx_conf_t, 147
mask
 vtss_vcap_ip_t, 298
 vtss_vcap_u128_t, 299
 vtss_vcap_u16_t, 300
 vtss_vcap_u24_t, 301
 vtss_vcap_u32_t, 302
 vtss_vcap_u40_t, 303
 vtss_vcap_u48_t, 304
 vtss_vcap_u8_t, 305
 vtss_vcap_vid_t, 308
 vtss_vcap_vr_t, 309

masquerade_port
 vtss_packet_tx_info_t, 179
master
 vtss_phy_conf_1g_t, 184
 vtss_phy_status_1g_t, 207
master_cfg_fault
 vtss_phy_status_1g_t, 206

max_frame_length
 vtss_port_conf_t, 221

max_length
 port_custom_conf_t, 25

max_tags
 port_custom_conf_t, 26
 vtss_port_conf_t, 222

mdi
 vtss_phy_conf_t, 186

mdi_cross
 vtss_port_status_t, 248

media_if
 vtss_phy_reset_conf_t, 202

media_if_pcs
 vtss_phy_conf_t, 187

media_mac_serdes_crc
 vtss_phy_statistic_t, 206

media_mac_serdes_good
 vtss_phy_statistic_t, 205

media_serdes_equipment_enable
 vtss_phy_loopback_t, 194

media_serdes_facility_enable
 vtss_phy_loopback_t, 194

media_serdes_input_enable
 vtss_phy_loopback_t, 193

miim_addr
 vtss_port_map_t, 235

miim_chip_no
 vtss_port_map_t, 235

miim_controller
 vtss_port_map_t, 235

miim_read
 vtss_init_conf_t, 112

miim_write
 vtss_init_conf_t, 112

mirror
 vtss_acl_action_t, 56
 vtss_vlan_vid_conf_t, 331

mld_cpu_only
 vtss_packet_rx_reg_t, 169

mmd_read
 vtss_init_conf_t, 112

mmd_read_inc
 vtss_init_conf_t, 112

mmd_write
 vtss_init_conf_t, 112

mode
 vtss_phy_conf_t, 185
 vtss_phy_led_mode_select_t, 191
 vtss_phy_power_conf_t, 200
 vtss_sgpio_port_conf_t, 283

vtss_ts_operation_mode_t, 295
 moved
 vtss_mac_table_status_t, 134
 mux_mode
 vtss_init_conf_t, 114

 nanoseconds
 vtss_timestamp_t, 291
 near_end_enable
 vtss_phy_loopback_t, 192
 network
 vtss_evc_conf_t, 89
 vtss_ipv4_uc_t, 120
 vtss_ipv6_uc_t, 122
 next
 tag_vtss_fdma_list, 30
 next_page
 vtss_port_clause_37_adv_t, 218
 nni
 vtss_evc_pb_conf_t, 92
 no_wait
 vtss_packet_rx_meta_t, 160
 now
 vtss_mtimer_t, 140
 npi
 vtss_packet_rx_queue_conf_t, 165
 number
 vtss_phy_led_mode_select_t, 191

 oam_info
 vtss_packet_rx_info_t, 158
 oam_info_decoded
 vtss_packet_rx_info_t, 158
 oam_type
 vtss_packet_tx_info_t, 177
 ob_post0
 serdes_fields_t, 27
 ob_sr
 serdes_fields_t, 27
 obey
 vtss_port_flow_control_conf_t, 228
 obey_pause
 vtss_aneg_t, 61
 one_pps_mode
 vtss_ts_ext_clock_mode_t, 292
 oper_up
 port_custom_conf_t, 26
 optimized_for_power
 vtss_eee_port_conf_t, 85
 options
 vtss_ace_frame_ipv4_t, 36
 vtss_vce_frame_ipv4_t, 313
 options.h
 VTSS_ARCH_CARACAL, 335
 VTSS_ARCH_LUTON26, 336
 VTSS_CHIP CU PHY, 349
 VTSS FEATURE ACL V2, 346
 VTSS FEATURE ACL, 346
 VTSS FEATURE AFI FDMA, 348

 VTSS FEATURE CLAUSE_37, 337
 VTSS FEATURE EEE, 345
 VTSS FEATURE EVC, 336
 VTSS FEATURE EXC_COL_CONT, 337
 VTSS FEATURE FAN, 345
 VTSS FEATURE FDMA, 348
 VTSS FEATURE INTERRUPTS, 348
 VTSS FEATURE IPV4_MC_SIP, 344
 VTSS FEATURE IPV6_MC_SIP, 344
 VTSS FEATURE IRQ_CONTROL, 347
 VTSS FEATURE LAYER2, 343
 VTSS FEATURE LED_POW_REDUC, 348
 VTSS FEATURE MAC_AGE_AUTO, 344
 VTSS FEATURE MAC_CPU_QUEUE, 345
 VTSS FEATURE MIRROR_CPU, 349
 VTSS FEATURE MISC, 336
 VTSS FEATURE NPI, 347
 VTSS FEATURE PACKET_GROUPING, 343
 VTSS FEATURE PACKET_PORT_REG, 343
 VTSS FEATURE PACKET_RX, 343
 VTSS FEATURE PACKET_TX, 342
 VTSS FEATURE PACKET, 342
 VTSS FEATURE PORT_CNT_BRIDGE, 337
 VTSS FEATURE PORT_CONTROL, 337
 VTSS FEATURE PORT_MUX, 345
 VTSS FEATURE PVLAN, 343
 VTSS FEATURE QCL_DMAC_DIP, 338
 VTSS FEATURE QCL_KEY_S_TAG, 338
 VTSS FEATURE QCL_PCP_DEI_ACTION, 338
 VTSS FEATURE QCL_POLICY_ACTION, 339
 VTSS FEATURE QCL_V2, 338
 VTSS FEATURE QCL, 338
 VTSS FEATURE QOS_CLASSIFICATION_V2, 341
 VTSS FEATURE QOS_DSCP_CLASS_DP_A↔WARE, 341
 VTSS FEATURE QOS_DSCP_REMARK_DP↔WARE, 342
 VTSS FEATURE QOS_DSCP_REMARK_V2, 342
 VTSS FEATURE QOS_DSCP_REMARK, 342
 VTSS FEATURE QOS_EGRESS_QUEUE_SH↔APERS_EB, 341
 VTSS FEATURE QOS_EGRESS_QUEUE_SH↔APERS, 341
 VTSS FEATURE QOS_POLICER_BC_SWITCH, 339
 VTSS FEATURE QOS_POLICER_DLB, 336
 VTSS FEATURE QOS_POLICER_MC_SWITCH, 339
 VTSS FEATURE QOS_POLICER_UC_SWITCH, 339
 VTSS FEATURE QOS_PORT_POLICER_EXT↔_FPS, 340
 VTSS FEATURE QOS_PORT_POLICER_EXT↔_FC, 340
 VTSS FEATURE QOS_PORT_POLICER_EXT, 339

VTSS_FEATURE_QOS_QUEUE_POLICER, 340
VTSS_FEATURE_QOS_QUEUE_TX, 340
VTSS_FEATURE_QOS_SCHEDULER_V2, 340
VTSS_FEATURE_QOS_TAG_REMARK_V2, 341
VTSS_FEATURE_QOS, 337
VTSS_FEATURE_SERDES_MACRO_SETTING_GS, 349
VTSS_FEATURE_SERIAL_GPIO, 336
VTSS_FEATURE_SFLOW, 349
VTSS_FEATURE_SYNC, 347
VTSS_FEATURE_TIMESTAMP_LATENCY_COUPLING_MP, 347
VTSS_FEATURE_TIMESTAMP_ONE_STEP, 346
VTSS_FEATURE_TIMESTAMP, 346
VTSS_FEATURE_VCAP, 345, 351
VTSS_FEATURE_VCL, 346
VTSS_FEATURE_VLAN_PORT_V2, 344
VTSS_FEATURE_VLAN_TRANSLATION, 348
VTSS_FEATURE_VLAN_TX_TAG, 344
VTSS_FEATURE_WARM_START, 351
VTSS_OPT_FDMA_DEBUG, 350
VTSS_OPT_FDMA_IRQ_CONTEXT, 350
VTSS_OPT_PORT_COUNT, 350
VTSS_OPT_TRACE, 349
VTSS_OPT_VAUI_EQ_CTRL, 350
VTSS_OPT_VCORE_III, 347
VTSS_PHY_OPT_VERIPHYS, 350
out_of_range
 vtss_rcpll_status_t, 275
outer_tag
 vtss_ece_action_t, 72

PORT_CAP_100M_FDX
 port.h, 356
PORT_CAP_100M_HDX
 port.h, 356
PORT_CAP_10G_FDX
 port.h, 357
PORT_CAP_10M_FDX
 port.h, 356
PORT_CAP_10M_HDX
 port.h, 356
PORT_CAP_1G_FDX
 port.h, 356
PORT_CAP_1G_PHY
 port.h, 361
PORT_CAP_2_5G_FDX
 port.h, 357
PORT_CAP_2_5G_TRI_SPEED_COPPER
 port.h, 364
PORT_CAP_2_5G_TRI_SPEED_FDX
 port.h, 364
PORT_CAP_2_5G_TRI_SPEED
 port.h, 364
PORT_CAP_5G_FDX
 port.h, 357
PORT_CAP_ANY_FIBER
 port.h, 362
PORT_CAP_AUTONEG
 port.h, 355
PORT_CAP_COPPER
 port.h, 357
PORT_CAP_DUAL_COPPER_100FX
 port.h, 360
PORT_CAP_DUAL_COPPER
 port.h, 358
PORT_CAP_DUAL_FIBER_1000X
 port.h, 363
PORT_CAP_DUAL_FIBER_100FX
 port.h, 359
PORT_CAP_DUAL_FIBER
 port.h, 358
PORT_CAP_DUAL_SFP_DETECT
 port.h, 360
PORT_CAP_FIBER
 port.h, 358
PORT_CAP_FLOW_CTRL
 port.h, 357
PORT_CAP_HDX
 port.h, 360
PORT_CAP_NONE
 port.h, 355
PORT_CAP_SD_ENABLE
 port.h, 358
PORT_CAP_SD_HIGH
 port.h, 358
PORT_CAP_SD_INTERNAL
 port.h, 359
PORT_CAP_SFP_1G
 port.h, 363
PORT_CAP_SFP_2_5G
 port.h, 364
PORT_CAP_SFP_DETECT
 port.h, 359
PORT_CAP_SFP_ONLY
 port.h, 360
PORT_CAP_SFP_SD_HIGH
 port.h, 364
PORT_CAP_SPEED_DUAL_ANY_FIBER_FIXED_SPEED
 port.h, 362
PORT_CAP_SPEED_DUAL_ANY_FIBER
 port.h, 363
PORT_CAP_STACKING
 port.h, 360
PORT_CAP_TRI_SPEED_COPPER
 port.h, 361
PORT_CAP_TRI_SPEED_DUAL_ANY_FIBER_FIXED_SPEED
 port.h, 363
PORT_CAP_TRI_SPEED_DUAL_ANY_FIBER
 port.h, 363
PORT_CAP_TRI_SPEED_DUAL_COPPER
 port.h, 362
PORT_CAP_TRI_SPEED_DUAL_FIBER_100FX
 port.h, 362
PORT_CAP_TRI_SPEED_DUAL_FIBER

port.h, 362
PORT_CAP_TRI_SPEED_FDX
 port.h, 361
PORT_CAP_TRI_SPEED_FIBER
 port.h, 361
PORT_CAP_TRI_SPEED
 port.h, 361
PORT_CAP_VTSS_10G_PHY
 port.h, 359
PORT_CAP_XAUI_LANE_FLIP
 port.h, 359
PRIi64
 types.h, 373
PRIu64
 types.h, 373
PRIx64
 types.h, 373
part_number
 vtss_chip_id_t, 64
 vtss_phy_type_t, 208
passwd
 vtss_secure_on_passwd_t, 279
pb
 vtss_evc_conf_t, 89
pcp
 vtss_ece_outer_tag_t, 80
 vtss_ece_tag_t, 82
 vtss_evc_inner_tag_t, 91
 vtss_qce_action_t, 251
 vtss_qce_tag_t, 263
 vtss_vce_tag_t, 322
 vtss_vlan_tag_t, 328
pcp_dei_enable
 vtss_qce_action_t, 250
pcp_dei_preserve
 vtss_ece_outer_tag_t, 80
 vtss_evc_inner_tag_t, 91
pd_enable
 vtss_phy_mac_serdes_pcs_ctrl_t, 195
pdu_offset
 vtss_packet_tx_info_t, 179
pfc
 port_custom_conf_t, 24
 vtss_port_flow_control_conf_t, 229
phy.h
 VTSS_PHY_POWER_ACTIPHY_BIT, 352
 VTSS_PHY_POWER_DYNAMIC_BIT, 352
 vtss_phy_power_mode_t, 352
 vtss_phy_veriphy_status_t, 353
phy_api_base_no
 vtss_phy_type_t, 209
pi
 vtss_init_conf_t, 114
pkt_mode
 vtss_phy_reset_conf_t, 202
police
 vtss_acl_action_t, 55
policer_bc
 vtss_qos_conf_t, 268
 policer_bc_frame_rate
 vtss_qos_conf_t, 268
policer_bc_mode
 vtss_qos_conf_t, 268
policer_ext_port
 vtss_qos_port_conf_t, 269
policer_mc
 vtss_qos_conf_t, 267
policer_mc_frame_rate
 vtss_qos_conf_t, 267
policer_mc_mode
 vtss_qos_conf_t, 267
policer_no
 vtss_acl_action_t, 55
policer_port
 vtss_qos_port_conf_t, 269
policer_queue
 vtss_qos_port_conf_t, 270
policer_uc
 vtss_qos_conf_t, 266
policer_uc_frame_rate
 vtss_qos_conf_t, 267
policer_uc_mode
 vtss_qos_conf_t, 267
policy
 vtss_ace_t, 49
policy_no
 vtss_acl_port_conf_t, 58
 vtss_ece_action_t, 72
 vtss_mce_action_t, 135
 vtss_qce_action_t, 251
 vtss_vce_action_t, 311
policy_no_enable
 vtss_qce_action_t, 251
pop_cnt
 vtss_mce_action_t, 135
pop_tag
 vtss_ece_action_t, 71
port.h
 PORT_CAP_100M_FDX, 356
 PORT_CAP_100M_HDX, 356
 PORT_CAP_10G_FDX, 357
 PORT_CAP_10M_FDX, 356
 PORT_CAP_10M_HDX, 356
 PORT_CAP_1G_FDX, 356
 PORT_CAP_1G_PHY, 361
 PORT_CAP_2_5G_FDX, 357
 PORT_CAP_2_5G_TRI_SPEED_COPPER, 364
 PORT_CAP_2_5G_TRI_SPEED_FDX, 364
 PORT_CAP_2_5G_TRI_SPEED, 364
 PORT_CAP_5G_FDX, 357
 PORT_CAP_ANY_FIBER, 362
 PORT_CAP_AUTONEG, 355
 PORT_CAP_COPPER, 357
 PORT_CAP_DUAL_COPPER_100FX, 360
 PORT_CAP_DUAL_COPPER, 358
 PORT_CAP_DUAL_FIBER_1000X, 363

PORT_CAP_DUAL_FIBER_100FX, 359
PORT_CAP_DUAL_FIBER, 358
PORT_CAP_DUAL_SFP_DETECT, 360
PORT_CAP_FIBER, 358
PORT_CAP_FLOW_CTRL, 357
PORT_CAP_HDX, 360
PORT_CAP_NONE, 355
PORT_CAP_SD_ENABLE, 358
PORT_CAP_SD_HIGH, 358
PORT_CAP_SD_INTERNAL, 359
PORT_CAP_SFP_1G, 363
PORT_CAP_SFP_2_5G, 364
PORT_CAP_SFP_DETECT, 359
PORT_CAP_SFP_ONLY, 360
PORT_CAP_SFP_SD_HIGH, 364
PORT_CAP_SPEED_DUAL_ANY_FIBER_FIXED_SPEED, 362
PORT_CAP_SPEED_DUAL_ANY_FIBER, 363
PORT_CAP_STACKING, 360
PORT_CAP_TRI_SPEED_COPPER, 361
PORT_CAP_TRI_SPEED_DUAL_ANY_FIBER_FIXED_SFP_SPEED, 363
PORT_CAP_TRI_SPEED_DUAL_ANY_FIBER, 363
PORT_CAP_TRI_SPEED_DUAL_COPPER, 362
PORT_CAP_TRI_SPEED_DUAL_FIBER_100FX, 362
PORT_CAP_TRI_SPEED_DUAL_FIBER, 362
PORT_CAP_TRI_SPEED_FDX, 361
PORT_CAP_TRI_SPEED_FIBER, 361
PORT_CAP_TRI_SPEED, 361
PORT_CAP_VTSS_10G_PHY, 359
PORT_CAP_XAUI_LANE_FLIP, 359
port_cap_t, 365
vtss_fiber_port_speed_t, 365
vtss_port_speed_t, 365
port_action
 vtss_acl_action_t, 56
port_cap_t
 port.h, 365
port_cnt
 vtss_phy_type_t, 209
port_conf
 vtss_sgpio_conf_t, 282
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, 26
 oper_up, 26
 pfc, 24
 power_mode, 25
speed, 24
port_list
 vtss_ace_t, 49
 vtss_acl_action_t, 56
 vtss_debug_info_t, 66
 vtss_ece_key_t, 77
 vtss_mce_key_t, 136
 vtss_qce_key_t, 258
 vtss_vce_key_t, 317
port_mask
 vtss_ts_timestamp_alloc_t, 296
port_no
 vtss_eps_port_conf_t, 88
 vtss_mirror_conf_t, 139
 vtss_npi_conf_t, 141
 vtss_packet_frame_info_t, 143
 vtss_packet_port_info_t, 146
 vtss_packet_rx_header_t, 149
 vtss_packet_rx_info_t, 151
 vtss_sync_clock_in_t, 286
port_tx
 vtss_packet_frame_info_t, 143
port_type
 vtss_vlan_port_conf_t, 326
ports
 vtss_vlan_trans_port2grp_conf_t, 330
power_down
 vtss_port_conf_t, 221
power_mode
 port_custom_conf_t, 25
ppr
 vtss_fan_conf_t, 95
pre_cb
 vtss_fdma_tx_info_t, 108
pre_cb_ctxt1
 vtss_fdma_tx_info_t, 108
pre_cb_ctxt2
 vtss_fdma_tx_info_t, 108
prefix_size
 vtss_ip_network_t, 118
 vtss_ipv4_network_t, 119
 vtss_ipv6_network_t, 121
prev_version
 vtss_restart_status_t, 276
prio
 vtss_ece_action_t, 72
 vtss_fdma_ch_cfg_t, 104
 vtss_mce_action_t, 135
 vtss_qce_action_t, 249
prio_enable
 vtss_ece_action_t, 72
 vtss_mce_action_t, 135
 vtss_qce_action_t, 249
prios
 vtss_qos_conf_t, 264
prop
 vtss_port_counters_t, 224
proto

vtss_ace_frame_ipv4_t, 36
 vtss_ace_frame_ipv6_t, 41
 vtss_ece_frame_ipv4_t, 74
 vtss_ece_frame_ipv6_t, 75
 vtss_qce_frame_ipv4_t, 253
 vtss_qce_frame_ipv6_t, 255
 vtss_vce_frame_ipv4_t, 313
 vtss_vce_frame_ipv6_t, 314
ptp
 vtss_ace_frame_etype_t, 35
 vtss_ace_frame_ipv4_t, 39
 vtss_ace_frame_ipv6_t, 44
ptp_action
 vtss_acl_action_t, 56
 vtss_packet_tx_info_t, 175
ptp_id
 vtss_packet_tx_info_t, 175
ptp_timestamp
 vtss_packet_tx_info_t, 176
pvid
 vtss_vlan_port_conf_t, 326
qos_class_map
 vtss_qos_port_conf_t, 271
qsgmii
 vtss_serdes_macro_conf_t, 280
queue
 vtss_packet_rx_conf_t, 147
queue_mask
 vtss_packet_rx_header_t, 149
queue_pct
 vtss_qos_port_conf_t, 274
r
 vtss_vcap_vr_t, 310
rate
 vtss_acl_policer_conf_t, 57
 vtss_policer_t, 215
 vtss_shaper_t, 285
raw_ident
 vtss_irq_status_t, 124
raw_mask
 vtss_irq_status_t, 125
raw_status
 vtss_irq_status_t, 125
reg
 vtss_packet_rx_conf_t, 147
reg_read
 vtss_init_conf_t, 111
reg_write
 vtss_init_conf_t, 111
remote_fault
 vtss_phy_media_serd_pcs_ctrl_t, 198
 vtss_port_clause_37_adv_t, 217
 vtss_port_status_t, 247
replaced
 vtss_mac_table_status_t, 133
req
 vtss_ace_frame_arp_t, 31
restart
 vtss_phy_mac_serd_pcs_ctrl_t, 195
 vtss_restart_status_t, 276
restart_info_port
 vtss_init_conf_t, 114
restart_info_src
 vtss_init_conf_t, 113
revision
 vtss_chip_id_t, 64
 vtss_phy_type_t, 208
rgmii
 vtss_phy_reset_conf_t, 202
rgmii_skew_delay_psec_t
 vtss_phy_api.h, 608
rmon
 vtss_port_counters_t, 224
route
 vtss_routing_entry_t, 278
rx_alloc_cb
 vtss_fdma_cfg_t, 97
rx_allow_multiple_dcbs
 vtss_fdma_cfg_t, 99
rx_allow_vlan_tag_mismatch
 vtss_fdma_cfg_t, 99
rx_buf_cnt
 vtss_fdma_cfg_t, 97
rx_cb
 vtss_fdma_cfg_t, 99
rx_clk_skew_ps
 vtss_phy_rgmii_conf_t, 203
rx_dont_reinsert_vlan_tag
 vtss_fdma_cfg_t, 99
rx_dont_strip_vlan_tag
 vtss_fdma_cfg_t, 98
rx_err_cnt_base_tx
 vtss_phy_statistic_t, 205
rx_etherStatsBroadcastPkts
 vtss_port_rmon_counters_t, 238
rx_etherStatsCRCAlignErrors
 vtss_port_rmon_counters_t, 238
rx_etherStatsDropEvents
 vtss_port_rmon_counters_t, 237
rx_etherStatsFragments
 vtss_port_rmon_counters_t, 239
rx_etherStatsJabbers
 vtss_port_rmon_counters_t, 239
rx_etherStatsMulticastPkts
 vtss_port_rmon_counters_t, 238
rx_etherStatsOctets
 vtss_port_rmon_counters_t, 237
rx_etherStatsOversizePkts
 vtss_port_rmon_counters_t, 239
rx_etherStatsPkts
 vtss_port_rmon_counters_t, 238
rx_etherStatsPkts1024to1518Octets
 vtss_port_rmon_counters_t, 240
rx_etherStatsPkts128to255Octets
 vtss_port_rmon_counters_t, 240

rx_etherStatsPkts1519toMaxOctets
 vtss_port_rmon_counters_t, 240

rx_etherStatsPkts256to511Octets
 vtss_port_rmon_counters_t, 240

rx_etherStatsPkts512to1023Octets
 vtss_port_rmon_counters_t, 240

rx_etherStatsPkts64Octets
 vtss_port_rmon_counters_t, 239

rx_etherStatsPkts65to127Octets
 vtss_port_rmon_counters_t, 239

rx_etherStatsUndersizePkts
 vtss_port_rmon_counters_t, 238

rx_frames
 vtss_basic_counters_t, 63

rx_green
 vtss_port_evc_counters_t, 226

rx_green_discard
 vtss_port_evc_counters_t, 227

rx_info
 tag_vtss_fdma_list, 29

rx_mtu
 vtss_fdma_cfg_t, 97

rx_prio
 vtss_port_proprietary_counters_t, 236

rx_red
 vtss_port_evc_counters_t, 227

rx_yellow
 vtss_port_evc_counters_t, 227

rx_yellow_discard
 vtss_port_evc_counters_t, 227

s_etype
 vtss_vlan_conf_t, 325

s_tag
 vtss_qce_tag_t, 263
 vtss_vce_tag_t, 323

s_tagged
 vtss_ece_tag_t, 83

sampling_rate
 vtss_sflow_port_conf_t, 281

sd_active_high
 vtss_port_conf_t, 220

sd_enable
 vtss_port_conf_t, 220

sd_internal
 vtss_port_conf_t, 220

sec
 vtss_timeofday_t, 289, 290

sec_msb
 vtss_timestamp_t, 290

seconds
 vtss_timestamp_t, 291

secure_on_enable
 vtss_phy_wol_conf_t, 211

select
 vtss_eee_port_state_t, 87

seq_zero
 vtss_ace_frame_ipv4_t, 39
 vtss_ace_frame_ipv6_t, 44

ser_led_frame_rate
 vtss_phy_enhanced_led_control_t, 189

ser_led_output_1
 vtss_phy_enhanced_led_control_t, 189

ser_led_output_2
 vtss_phy_enhanced_led_control_t, 189

ser_led_select
 vtss_phy_enhanced_led_control_t, 189

serdes
 vtss_init_conf_t, 114
 vtss_port_conf_t, 223

serdes1g_vdd
 vtss_serdes_macro_conf_t, 279

serdes6g_vdd
 vtss_serdes_macro_conf_t, 280

serdes_aneg_ena
 vtss_phy_mac_serd_pcs_cntl_t, 196

serdes_fields_t, 26
 ob_post0, 27
 ob_sr, 27

serdes_pol_inv_in
 vtss_phy_mac_serd_pcs_cntl_t, 196
 vtss_phy_media_serd_pcs_cntl_t, 198

serdes_pol_inv_out
 vtss_phy_mac_serd_pcs_cntl_t, 196
 vtss_phy_media_serd_pcs_cntl_t, 198

serdes_tx_bad
 vtss_phy_statistic_t, 205

serdes_tx_good
 vtss_phy_statistic_t, 205

sflow_port_no
 vtss_packet_rx_info_t, 157

sflow_queue
 vtss_packet_rx_queue_map_t, 167

sflow_type
 vtss_packet_rx_info_t, 157

sfp_dac
 vtss_port_serdes_conf_t, 244

sgmii_in_pre
 vtss_phy_mac_serd_pcs_cntl_t, 196

sgmii_out_pre
 vtss_phy_mac_serd_pcs_cntl_t, 196

shaper_port
 vtss_qos_port_conf_t, 270

shaper_queue
 vtss_qos_port_conf_t, 270

sigdet
 vtss_phy_conf_t, 186

sip
 vtss_ace_frame_arp_t, 33
 vtss_ace_frame_ipv4_t, 37
 vtss_ace_frame_ipv6_t, 41
 vtss_ace_sip_smac_t, 48
 vtss_ece_frame_ipv4_t, 74
 vtss_ece_frame_ipv6_t, 75
 vtss_qce_frame_ipv4_t, 253
 vtss_qce_frame_ipv6_t, 255
 vtss_vce_frame_ipv4_t, 313

vtss_vce_frame_ipv6_t, 315
 sip_dip_enable
 vtss_aggr_mode_t, 60
 sip_eq_dip
 vtss_ace_frame_ipv4_t, 39
 vtss_ace_frame_ipv6_t, 43
 sip_smac
 vtss_ace_frame_ipv4_t, 40
 size
 vtss_packet_rx_queue_conf_t, 164
 skip_coma
 vtss_phy_conf_t, 187
 smac
 vtss_ace_frame_arp_t, 31
 vtss_ace_frame_etype_t, 34
 vtss_ace_frame_llc_t, 45
 vtss_ace_frame_snap_t, 46
 vtss_ace_sip_smac_t, 48
 vtss_ece_mac_t, 79
 vtss_port_flow_control_conf_t, 229
 vtss_qce_mac_t, 261
 vtss_vce_mac_t, 320
 smac_enable
 vtss_aggr_mode_t, 59
 smac_match
 vtss_ace_frame_arp_t, 32
 snap
 vtss_ace_frame_snap_t, 46
 vtss_ace_t, 51
 vtss_qce_key_t, 259
 vtss_vce_key_t, 318
 speed
 port_custom_conf_t, 24
 vtss_phy_forced_t, 190
 vtss_port_conf_t, 221
 vtss_port_status_t, 247
 speed_100M
 vtss_port_sgmii_aneg_t, 245
 speed_100m_fdx
 vtss_phy_aneg_t, 181
 speed_100m_hdx
 vtss_phy_aneg_t, 181
 speed_10M
 vtss_port_sgmii_aneg_t, 245
 speed_10m_fdx
 vtss_phy_aneg_t, 181
 speed_10m_hdx
 vtss_phy_aneg_t, 181
 speed_1G
 vtss_port_sgmii_aneg_t, 245
 speed_1g_fdx
 vtss_phy_aneg_t, 181
 speed_1g_hdx
 vtss_phy_aneg_t, 181
 spi_32bit_read_write
 vtss_init_conf_t, 113
 spi_64bit_read_write
 vtss_init_conf_t, 113
 spi_read_write
 vtss_init_conf_t, 113
 sport
 vtss_ace_frame_ipv4_t, 37
 vtss_ace_frame_ipv6_t, 42
 vtss_ece_frame_ipv4_t, 74
 vtss_ece_frame_ipv6_t, 76
 vtss_qce_frame_ipv4_t, 254
 vtss_qce_frame_ipv6_t, 255
 sport_dport_enable
 vtss_aggr_mode_t, 60
 sport_eq_dport
 vtss_ace_frame_ipv4_t, 39
 vtss_ace_frame_ipv6_t, 43
 squelch
 vtss_phy_clock_conf_t, 183
 squelsh
 vtss_sync_clock_in_t, 286
 src
 vtss_phy_clock_conf_t, 183
 stack_queue
 vtss_packet_rx_queue_map_t, 167
 status
 vtss_phy_veriphy_result_t, 210
 stripped_tag
 vtss_packet_rx_info_t, 153
 suspend_tick_cnt
 vtss_fdma_throttle_cfg_t, 107
 sw_tstamp
 tag_vtss_fdma_list, 29
 vtss_packet_rx_info_t, 155
 vtss_packet_rx_meta_t, 162
 switch_frm
 vtss_packet_tx_info_t, 172
 symmetric_pause
 vtss_phy_aneg_t, 182
 vtss_port_clause_37_adv_t, 217
 TRUE
 types.h, 375
 tag
 vtss_ece_key_t, 77
 vtss_packet_rx_header_t, 149
 vtss_packet_rx_info_t, 153
 vtss_packet_tx_info_t, 173
 vtss_qce_key_t, 258
 vtss_vce_key_t, 317
 tag_class_enable
 vtss_qos_port_conf_t, 271
 tag_default_dei
 vtss_qos_port_conf_t, 273
 tag_default_pcp
 vtss_qos_port_conf_t, 273
 tag_dei_map
 vtss_qos_port_conf_t, 273
 tag_pcp_map
 vtss_qos_port_conf_t, 273
 tag_remark_mode
 vtss_qos_port_conf_t, 273

tag_type
 vtss_packet_rx_info_t, 152

tag_vtss_fdma_list, 27
 act_len, 28
 afi_frm_cnt, 30
 afi_seq_number, 30
 alloc_ptr, 29
 frm_ptr, 28
 next, 30
 rx_info, 29
 sw_tstamp, 29
 user, 29

tagged
 vtss_ace_vlan_t, 53
 vtss_ece_tag_t, 83
 vtss_qce_tag_t, 263
 vtss_vce_tag_t, 322

tagprio
 vtss_tci_t, 289

target
 vtss_inst_create_t, 115

tbi
 vtss_phy_reset_conf_t, 202

tcp_ack
 vtss_ace_frame_ipv4_t, 38
 vtss_ace_frame_ipv6_t, 43

tcp_fin
 vtss_ace_frame_ipv4_t, 38
 vtss_ace_frame_ipv6_t, 42

tcp_psh
 vtss_ace_frame_ipv4_t, 38
 vtss_ace_frame_ipv6_t, 43

tcp_RST
 vtss_ace_frame_ipv4_t, 38
 vtss_ace_frame_ipv6_t, 42

tcp_SYN
 vtss_ace_frame_ipv4_t, 38
 vtss_ace_frame_ipv6_t, 42

tcp_URG
 vtss_ace_frame_ipv4_t, 39
 vtss_ace_frame_ipv6_t, 43

timeout
 vtss_mtimer_t, 140

tod_get_ns_cnt_cb_t
 vtss_misc_api.h, 511

tpid
 vtss_packet_port_filter_t, 145
 vtss_vlan_tag_t, 328

trans_vid
 vtss_vlan_trans_grp2vlan_conf_t, 329

ts
 vtss_ts_timestamp_t, 297

ts_id
 vtss_ts_id_t, 293

ts_valid
 vtss_ts_timestamp_t, 297

tstamp_id
 vtss_packet_rx_info_t, 156

tstamp_id_decoded
 vtss_packet_rx_info_t, 156

tt_loop
 vtss_mce_t, 137

ttl
 vtss_ace_frame_ipv4_t, 36
 vtss_ace_frame_ipv6_t, 41

tx_buf_cnt
 vtss_fdma_cfg_t, 100

tx_clk_skew_ps
 vtss_phy_rgmii_conf_t, 204

tx_done_cb
 vtss_fdma_cfg_t, 100

tx_etherStatsBroadcastPkts
 vtss_port_rmon_counters_t, 241

tx_etherStatsCollisions
 vtss_port_rmon_counters_t, 242

tx_etherStatsDropEvents
 vtss_port_rmon_counters_t, 241

tx_etherStatsMulticastPkts
 vtss_port_rmon_counters_t, 241

tx_etherStatsOctets
 vtss_port_rmon_counters_t, 241

tx_etherStatsPkts
 vtss_port_rmon_counters_t, 241

tx_etherStatsPkts1024to1518Octets
 vtss_port_rmon_counters_t, 243

tx_etherStatsPkts128to255Octets
 vtss_port_rmon_counters_t, 242

tx_etherStatsPkts1519toMaxOctets
 vtss_port_rmon_counters_t, 243

tx_etherStatsPkts256to511Octets
 vtss_port_rmon_counters_t, 242

tx_etherStatsPkts512to1023Octets
 vtss_port_rmon_counters_t, 243

tx_etherStatsPkts64Octets
 vtss_port_rmon_counters_t, 242

tx_frames
 vtss_basic_counters_t, 63

tx_green
 vtss_port_evc_counters_t, 227

tx_out_bytes
 vtss_eee_port_counter_t, 86

tx_out_bytes_get
 vtss_eee_port_counter_t, 86

tx_prio
 vtss_port_proprietary_counters_t, 236

tx_remote_fault
 vtss_phy_aneg_t, 182

tx_tw
 vtss_eee_port_conf_t, 84

tx_yellow
 vtss_port_evc_counters_t, 228

type
 vtss_ace_t, 50
 vtss_dlb_policer_conf_t, 69

vtss_ece_key_t, 77
 vtss_eps_port_conf_t, 88
 vtss_evc_inner_tag_t, 90
 vtss_fan_conf_t, 95
 vtss_ip_addr_t, 116
 vtss_qce_key_t, 258
 vtss_routing_entry_t, 277
 vtss_sflow_port_conf_t, 281
 vtss_vcap_vr_t, 309
 vtss_vce_key_t, 318
types.h
 BOOL, 393
 FALSE, 375
 i16, 392
 i32, 392
 i64, 392
 i8, 392
 MAC_ADDR_BROADCAST, 383
 PRIi64, 373
 PRIu64, 373
 PRIx64, 373
 TRUE, 375
 u16, 393
 u32, 393
 u64, 393
 u8, 392
 uintptr_t, 393
 VTSS_ACL_POLICER_NO_END, 387
 VTSS_ACL_POLICER_NO_START, 387
 VTSS_ACL_POLICERS, 387
 VTSS_ACL_POLICIES, 388
 VTSS_ACL_POLICY_NO_END, 389
 VTSS_ACL_POLICY_NO_MAX, 388
 VTSS_ACL_POLICY_NO_MIN, 388
 VTSS_ACL_POLICY_NO_NONE, 388
 VTSS_ACL_POLICY_NO_START, 388
 VTSS_AGGR_NO_END, 384
 VTSS_AGGR_NO_NONE, 384
 VTSS_AGGR_NO_START, 384
 VTSS_AGGRS, 383
 VTSS_BIT64, 374
 VTSS_BITMASK64, 374
 VTSS_BITRATE_DISABLED, 381
 VTSS_CLOCK_IDENTITY_LENGTH, 391
 VTSS_DEI_ARRAY_SIZE, 380
 VTSS_DEI_END, 380
 VTSS_DEI_START, 380
 VTSS_DEIS, 380
 VTSS_DPL_ARRAY_SIZE, 381
 VTSS_DPL_END, 381
 VTSS_DPL_START, 381
 VTSS_DPLS, 380
 VTSS_ENCODE_BITFIELD64, 374
 VTSS_ENCODE_BITMASK64, 374
 VTSSETYPE_VTSS, 382
 VTSS_EVCS, 383
 VTSS_EXTRACT_BITFIELD64, 374
 VTSS_GLAG_NO_END, 385
 VTSS_GLAG_NO_NONE, 384
 VTSS_GLAG_NO_START, 385
 VTSS_GLAG_PORT_ARRAY_SIZE, 386
 VTSS_GLAG_PORT_END, 385
 VTSS_GLAG_PORT_START, 385
 VTSS_GLAG_PORTS, 385
 VTSS_GLAGS, 384
 VTSS_HQOS_COUNT, 389
 VTSS_HQOS_ID_NONE, 389
 VTSS_INTERVAL_MS, 390
 VTSS_INTERVAL_NS, 390
 VTSS_INTERVAL_PS, 391
 VTSS_INTERVAL_SEC, 390
 VTSS_INTERVAL_US, 390
 VTSS_ISDX_NONE, 383
 VTSS_MAC_ADDR_SZ_BYTES, 383
 VTSS_MAX_TIMEINTERVAL, 390
 VTSS_ONE_MILL, 389
 VTSS_ONE_MIA, 389
 VTSS_PACKET_RATE_DISABLED, 375
 VTSS_PACKET_RX_GRP_CNT, 386
 VTSS_PACKET_RX_QUEUE_CNT, 386
 VTSS_PACKET_RX_QUEUE_END, 387
 VTSS_PACKET_RX_QUEUE_NONE, 386
 VTSS_PACKET_RX_QUEUE_START, 387
 VTSS_PACKET_TX_GRP_CNT, 386
 VTSS_PCP_ARRAY_SIZE, 379
 VTSS_PCP_END, 379
 VTSS_PCP_START, 379
 VTSS_PCPS, 379
 VTSS_PORT_ARRAY_SIZE, 377
 VTSS_PORT_COUNT, 375
 VTSS_PORT_IS_PORT, 377
 VTSS_PORT_NO_CPU, 376
 VTSS_PORT_NO_END, 376
 VTSS_PORT_NO_NONE, 376
 VTSS_PORT_NO_START, 376
 VTSS_PORTS, 376
 VTSS_PRIO_ARRAY_SIZE, 378
 VTSS_PRIO_END, 378
 VTSS_PRIO_NO_NONE, 377
 VTSS_PRIO_START, 377
 VTSS_PRIOS, 377
 VTSS_QUEUE_ARRAY_SIZE, 379
 VTSS_QUEUE_END, 378
 VTSS_QUEUE_START, 378
 VTSS_QUEUES, 378
 VTSS_SEC_NS_INTERVAL, 391
 VTSS_SYNCE_CLK_PORT_ARRAY_SIZE, 391
 VTSS_VID_ALL, 382
 VTSS_VID_DEFAULT, 382
 VTSS_VID_NULL, 381
 VTSS_VID_RESERVED, 382
 VTSS_VIDS, 382
 vtss_ece_dir_t, 401
 vtss_ece_pop_tag_t, 402
 vtss_hqos_sch_mode_t, 402
 vtss_ip_type_t, 400

vtss_isdx_t, 394
vtss_mac_addr_t, 394
vtss_mem_flags_t, 397
vtss_packet_reg_type_t, 399
vtss_packet_rx_grp_t, 394
vtss_packet_tx_grp_t, 394
vtss_policer_type_t, 399
vtss_port_interface_t, 397
vtss_serdes_mode_t, 398
vtss_storm_policer_mode_t, 399
vtss_vcap_bit_t, 400
vtss_vcap_key_type_t, 401
vtss_vcap_vr_type_t, 401
vtss_vdd_t, 400
vtss_vlan_frame_t, 399

u16
types.h, 393

u32
types.h, 393

u64
types.h, 393

u8
types.h, 392

uint
vtss_os_custom.h, 541

uintptr_t
types.h, 393

ulong
vtss_os_custom.h, 541

unidir
vtss_phy_conf_t, 186

unidirectional_ability
vtss_port_status_t, 248

unknown
vtss_ace_frame_arp_t, 32

untagged_vid
vtss_vlan_port_conf_t, 326

usage
vtss_fdma_ch_cfg_t, 102

use_extended_bus_cycle
vtss_pi_conf_t, 213

user
tag_vtss_fdma_list, 29

usr_prio
vtss_ace_vlan_t, 53
vtss_qos_port_conf_t, 271

uvvid
vtss_evc_pb_conf_t, 92

v
vtss_vcap_vr_t, 309

VTSS_ACE_ID_LAST
vtss_security_api.h, 686

VTSS_ACL_POLICER_NO_END
types.h, 387

VTSS_ACL_POLICER_NO_START
types.h, 387

VTSS_ACL_POLICERS

types.h, 387
VTSS_ACL_POLICIES
types.h, 388

VTSS_ACL_POLICY_NO_END
types.h, 389

VTSS_ACL_POLICY_NO_MAX
types.h, 388

VTSS_ACL_POLICY_NO_MIN
types.h, 388

VTSS_ACL_POLICY_NO_NONE
types.h, 388

VTSS_ACL_POLICY_NO_START
types.h, 388

VTSS_AFI_FPS_MAX
vtss_fdma_api.h, 419

VTSS_AGGR_NO_END
types.h, 384

VTSS_AGGR_NO_NONE
types.h, 384

VTSS_AGGR_NO_START
types.h, 384

VTSS_AGGRS
types.h, 383

VTSS_ARCH_CARACAL
options.h, 335

VTSS_ARCH_LUTON26
options.h, 336

VTSS_BIT64
types.h, 374

VTSS_BITMASK64
types.h, 374

VTSS_BITRATE_DISABLED
types.h, 381

VTSS_CHIP CU PHY
options.h, 349

VTSS_CLOCK_IDENTITY_LENGTH
types.h, 391

VTSS_DEI_ARRAY_SIZE
types.h, 380

VTSS_DEI_END
types.h, 380

VTSS_DEI_START
types.h, 380

VTSS_DEIS
types.h, 380

VTSS_DIV64
vtss_os_custom.h, 542
vtss_os_ecos.h, 547
vtss_os_linux.h, 559

VTSS_DPL_ARRAY_SIZE
types.h, 381

VTSS_DPL_END
types.h, 381

VTSS_DPL_START
types.h, 381

VTSS_DPLS
types.h, 380

VTSS_ECE_ID_LAST

vtss_evc_api.h, 407
VTSS_ENCODE_BITFIELD64
 types.h, 374
VTSS_ENCODE_BITMASK64
 types.h, 374
VTSS_ERPI_ARRAY_SIZE
 vtss_l2_api.h, 460
VTSS_ERPI_END
 vtss_l2_api.h, 460
VTSS_ERPI_START
 vtss_l2_api.h, 459
VTSS_ERPIS
 vtss_l2_api.h, 459
VTSSETYPE_VTSS
 types.h, 382
VTSS_EVC_ID_NONE
 vtss_evc_api.h, 406
VTSS_EVC_POLICERS
 vtss_evc_api.h, 406
VTSS_EVCS
 types.h, 383
VTSS_EXTRACT_BITFIELD64
 types.h, 374
VTSS_FDMA_CCM_FPS_MAX
 vtss_fdma_api.h, 419
VTSS_FDMA_CCM_FREQ_LIST_LEN
 vtss_fdma_api.h, 419
VTSS_FDMA_CCM_QUOTIENT_MAX
 vtss_fdma_api.h, 419
VTSS_FDMA_CH_CNT
 vtss_fdma_api.h, 416
VTSS_FDMA_DC布_SIZE_BYTES
 vtss_fdma_api.h, 417
VTSS_FDMA_HDR_SIZE_BYTES
 vtss_fdma_api.h, 417
VTSS_FDMA_MAX_DATA_PER_DC布_BYTES
 vtss_fdma_api.h, 417
VTSS_FDMA_MAX_FRAME_SIZE_BYTES
 vtss_fdma_api.h, 418
VTSS_FDMA_MIN_DATA_PER_INJ_SOF_DC布_BYTES
 TES
 vtss_fdma_api.h, 417
VTSS_FDMA_MIN_DATA_PER_NON_SOF_DC布_BYTES
 YTES
 vtss_fdma_api.h, 418
VTSS_FDMA_MIN_DATA_PER_XTR_SOF_DC布_BYTES
 YTES
 vtss_fdma_api.h, 418
VTSS_FDMA_MIN_FRAME_SIZE_BYTES
 vtss_fdma_api.h, 417
VTSS_FEATURE_ACL_V2
 options.h, 346
VTSS_FEATURE_ACL
 options.h, 346
VTSS_FEATURE_AFI_FDMA
 options.h, 348
VTSS_FEATURE_CLAUSE_37
 options.h, 337

VTSS_FEATURE_EEE
 options.h, 345
VTSS_FEATURE_EVC
 options.h, 336
VTSS_FEATURE_EXC_COL_CONT
 options.h, 337
VTSS_FEATURE_FAN
 options.h, 345
VTSS_FEATURE_FDMA
 options.h, 348
VTSS_FEATURE_INTERRUPTS
 options.h, 348
VTSS_FEATURE_IPV4_MC_SIP
 options.h, 344
VTSS_FEATURE_IPV6_MC_SIP
 options.h, 344
VTSS_FEATURE_IRQ_CONTROL
 options.h, 347
VTSS_FEATURE_LAYER2
 options.h, 343
VTSS_FEATURE_LED_POW_REDUC
 options.h, 348
VTSS_FEATURE_MAC_AGE_AUTO
 options.h, 344
VTSS_FEATURE_MAC_CPU_QUEUE
 options.h, 345
VTSS_FEATURE_MIRROR_CPU
 options.h, 349
VTSS_FEATURE_MISC
 options.h, 336
VTSS_FEATURE_NPI
 options.h, 347
VTSS_FEATURE_PACKET_GROUPING
 options.h, 343
VTSS_FEATURE_PACKET_PORT_REG
 options.h, 343
VTSS_FEATURE_PACKET_RX
 options.h, 343
VTSS_FEATURE_PACKET_TX
 options.h, 342
VTSS_FEATURE_PACKET
 options.h, 342
VTSS_FEATURE_PORT_CNT_BRIDGE
 options.h, 337
VTSS_FEATURE_PORT_CONTROL
 options.h, 337
VTSS_FEATURE_PORT_MUX
 options.h, 345
VTSS_FEATURE_PVLAN
 options.h, 343
VTSS_FEATURE_QCL_DMAC_DIP
 options.h, 338
VTSS_FEATURE_QCL_KEY_S_TAG
 options.h, 338
VTSS_FEATURE_QCL_PCP_DEI_ACTION
 options.h, 338
VTSS_FEATURE_QCL_POLICY_ACTION
 options.h, 339

VTSS_FEATURE_QCL_V2
 options.h, 338
VTSS_FEATURE_QCL
 options.h, 338
VTSS_FEATURE_QOS_CLASSIFICATION_V2
 options.h, 341
VTSS_FEATURE_QOS_DSCP_CLASS_DP_AWARE
 options.h, 341
VTSS_FEATURE_QOS_DSCP_REMARK_DP_AWA←
 RE
 options.h, 342
VTSS_FEATURE_QOS_DSCP_REMARK_V2
 options.h, 342
VTSS_FEATURE_QOS_DSCP_REMARK
 options.h, 342
VTSS_FEATURE_QOS_EGRESS_QUEUE_SHAPE←
 RS_EB
 options.h, 341
VTSS_FEATURE_QOS_EGRESS_QUEUE_SHAPE←
 RS
 options.h, 341
VTSS_FEATURE_QOS_POLICER_BC_SWITCH
 options.h, 339
VTSS_FEATURE_QOS_POLICER_DLBB
 options.h, 336
VTSS_FEATURE_QOS_POLICER_MC_SWITCH
 options.h, 339
VTSS_FEATURE_QOS_POLICER_UC_SWITCH
 options.h, 339
VTSS_FEATURE_QOS_PORT_POLICER_EXT_FPS
 options.h, 340
VTSS_FEATURE_QOS_PORT_POLICER_EXT_FC
 options.h, 340
VTSS_FEATURE_QOS_PORT_POLICER_EXT
 options.h, 339
VTSS_FEATURE_QOS_QUEUE_POLICER
 options.h, 340
VTSS_FEATURE_QOS_QUEUE_TX
 options.h, 340
VTSS_FEATURE_QOS_SCHEDULER_V2
 options.h, 340
VTSS_FEATURE_QOS_TAG_REMARK_V2
 options.h, 341
VTSS_FEATURE_QOS
 options.h, 337
VTSS_FEATURE_SERDES_MACRO_SETTINGS
 options.h, 349
VTSS_FEATURE_SERIAL_GPIO
 options.h, 336
VTSS_FEATURE_SFLOW
 options.h, 349
VTSS_FEATURE_SYNCE
 options.h, 347
VTSS_FEATURE_TIMESTAMP_LATENCY_COMP
 options.h, 347
VTSS_FEATURE_TIMESTAMP_ONE_STEP
 options.h, 346
VTSS_FEATURE_TIMESTAMP
 options.h, 346
VTSS_FEATURE_VCAP
 options.h, 345, 351
VTSS_FEATURE_VCL
 options.h, 346
VTSS_FEATURE_VLAN_PORT_V2
 options.h, 344
VTSS_FEATURE_VLAN_TRANSLATION
 options.h, 348
VTSS_FEATURE_VLAN_TX_TAG
 options.h, 344
VTSS_FEATURE_WARM_START
 options.h, 351
VTSS_FRAME_GAP_DEFAULT
 vtss_port_api.h, 664
VTSS_GLAG_NO_END
 types.h, 385
VTSS_GLAG_NO_NONE
 types.h, 384
VTSS_GLAG_NO_START
 types.h, 385
VTSS_GLAG_PORT_ARRAY_SIZE
 types.h, 386
VTSS_GLAG_PORT_END
 types.h, 385
VTSS_GLAG_PORT_START
 types.h, 385
VTSS_GLAG_PORTS
 types.h, 385
VTSS_GLGS
 types.h, 384
VTSS_HQOS_COUNT
 types.h, 389
VTSS_HQOS_ID_NONE
 types.h, 389
VTSS_I2C_NO_MULTIPLEXER
 vtss_init_api.h, 435
VTSS_INTERVAL_MS
 types.h, 390
VTSS_INTERVAL_NS
 types.h, 390
VTSS_INTERVAL_PS
 types.h, 391
VTSS_INTERVAL_SEC
 types.h, 390
VTSS_INTERVAL_US
 types.h, 390
VTSS_ISDX_NONE
 types.h, 383
VTSS_JR1_PACKET_HDR_SIZE_BYTES
 vtss_packet_api.h, 566
VTSS_JR1_RX_IFH_SIZE
 vtss_packet_api.h, 567
VTSS_JR2_PACKET_HDR_SIZE_BYTES
 vtss_packet_api.h, 566
VTSS_JR2_RX_IFH_SIZE
 vtss_packet_api.h, 567
VTSS_L26_PACKET_HDR_SIZE_BYTES

vtss_packet_api.h, 566
VTSS_L26_RX_IFH_SIZE
 vtss_packet_api.h, 567
VTSS_LABS
 vtss_os_custom.h, 542
 vtss_os_ecos.h, 548
 vtss_os_linux.h, 559
VTSS_LLabs
 vtss_os_custom.h, 543
 vtss_os_ecos.h, 548
 vtss_os_linux.h, 559
VTSS_MAC_ADDR_SZ_BYTES
 types.h, 383
VTSS_MAC_ADDRS
 vtss_l2_api.h, 454
VTSS_MAX_FRAME_LENGTH_MAX
 vtss_port_api.h, 665
VTSS_MAX_FRAME_LENGTH_STANDARD
 vtss_port_api.h, 664
VTSS_MAX_TIMEINTERVAL
 types.h, 390
VTSS_MCE_ID_LAST
 vtss_evc_api.h, 407
VTSS_MCE_POP_NONE
 vtss_evc_api.h, 407
VTSS_MOD64
 vtss_os_custom.h, 542
 vtss_os_ecos.h, 547
 vtss_os_linux.h, 559
VTSS_MSLEEP
 vtss_os_custom.h, 541
 vtss_os_ecos.h, 546
 vtss_os_linux.h, 556
VTSS_MSTI_ARRAY_SIZE
 vtss_l2_api.h, 454
VTSS_MSTI_END
 vtss_l2_api.h, 454
VTSS_MSTI_START
 vtss_l2_api.h, 454
VTSS_MSTIS
 vtss_l2_api.h, 454
VTSS_MTIMER_CANCEL
 vtss_os_custom.h, 542
 vtss_os_ecos.h, 547
 vtss_os_linux.h, 557
VTSS_MTIMER_START
 vtss_os_custom.h, 541
 vtss_os_ecos.h, 546
 vtss_os_linux.h, 557
VTSS_MTIMER_TIMEOUT
 vtss_os_custom.h, 541
 vtss_os_ecos.h, 547
 vtss_os_linux.h, 557
VTSS_NSLEEP
 vtss_os_ecos.h, 546
 vtss_os_linux.h, 556
VTSS_ONE_MILL
 types.h, 389
VTSS_ONE_MIA
 types.h, 389
VTSS_OPT_FDMA_DEBUG
 options.h, 350
VTSS_OPT_FDMA_IRQ_CONTEXT
 options.h, 350
VTSS_OPT_PORT_COUNT
 options.h, 350
VTSS_OPT_TRACE
 options.h, 349
VTSS_OPT_VAUI_EQ_CTRL
 options.h, 350
VTSS_OPT_VCORE_III
 options.h, 347
VTSS_OS_BIG_ENDIAN
 vtss_os_ecos.h, 551
 vtss_os_linux.h, 556
VTSS_OS_COMPILER_ATTRIBUTE_ALIGNED
 vtss_fdma_api.h, 418
 vtss_os_ecos.h, 550
VTSS_OS_CTZ64
 vtss_os_custom.h, 543
 vtss_os_ecos.h, 548
 vtss_os_linux.h, 561
VTSS_OS_CTZ
 vtss_os_custom.h, 543
 vtss_os_ecos.h, 548
 vtss_os_linux.h, 560
VTSS_OS_DCACHE_FLUSH
 vtss_os_ecos.h, 551
VTSS_OS_DCACHE_INVALIDATE
 vtss_os_ecos.h, 550
VTSS_OS_DCACHE_LINE_SIZE_BYTES
 vtss_fdma_api.h, 418
 vtss_os_ecos.h, 550
VTSS_OS_FREE
 vtss_os_custom.h, 544
 vtss_os_ecos.h, 549
 vtss_os_linux.h, 561
VTSS_OS_INTERRUPT_DISABLE
 vtss_os_ecos.h, 553
VTSS_OS_INTERRUPT_FLAGS
 vtss_os_ecos.h, 552
VTSS_OS_INTERRUPT_RESTORE
 vtss_os_ecos.h, 553
VTSS_OS_MALLOC
 vtss_os_custom.h, 543
 vtss_os_ecos.h, 549
 vtss_os_linux.h, 561
VTSS_OS_NTOHL
 vtss_os_ecos.h, 551
 vtss_os_linux.h, 556
VTSS_OS_RAND
 vtss_os_custom.h, 544
 vtss_os_ecos.h, 549
 vtss_os_linux.h, 562
VTSS_OS_reordered_barrier
 vtss_os_ecos.h, 550

VTSS_OS_SCHEDULER_FLAGS
vtss_os_ecos.h, 552
vtss_os_linux.h, 558
VTSS_OS_SCHEDULER_LOCK
vtss_os_ecos.h, 552
vtss_os_linux.h, 558
VTSS_OS_SCHEDULER_UNLOCK
vtss_os_ecos.h, 552
vtss_os_linux.h, 558
VTSS_OS_TIMESTAMP_TYPE
vtss_misc_api.h, 510
VTSS_OS_TIMESTAMP
vtss_misc_api.h, 510
VTSS_OS_VIRT_TO_PHYS
vtss_os_ecos.h, 551
VTSS_PACKET_HDR_SIZE_BYTES
vtss_packet_api.h, 566
VTSS_PACKET_RATE_DISABLED
types.h, 375
VTSS_PACKET_RX_GRP_CNT
types.h, 386
VTSS_PACKET_RX_QUEUE_CNT
types.h, 386
VTSS_PACKET_RX_QUEUE_END
types.h, 387
VTSS_PACKET_RX_QUEUE_NONE
types.h, 386
VTSS_PACKET_RX_QUEUE_START
types.h, 387
VTSS_PACKET_TX_GRP_CNT
types.h, 386
VTSS_PACKET_TX_IFH_MAX
vtss_packet_api.h, 567
VTSS_PCP_ARRAY_SIZE
types.h, 379
VTSS_PCP_END
types.h, 379
VTSS_PCP_START
types.h, 379
VTSS_PCPS
types.h, 379
VTSS_PHY_ACTIPHY_PWR
vtss_phy_api.h, 595
VTSS_PHY_LINK_AMS_EV
vtss_phy_api.h, 600
VTSS_PHY_LINK_AUTO_NEG_COMPLETE_EV
vtss_phy_api.h, 600
VTSS_PHY_LINK_AUTO_NEG_ERROR_EV
vtss_phy_api.h, 600
VTSS_PHY_LINK_DOWN_PWR
vtss_phy_api.h, 595
VTSS_PHY_LINK_EXT_EEE_LINKFAIL_EV
vtss_phy_api.h, 603
VTSS_PHY_LINK_EXT_EEE_WAIT_RX_TQ_EV
vtss_phy_api.h, 603
VTSS_PHY_LINK_EXT_EEE_WAIT_TS_EV
vtss_phy_api.h, 603
VTSS_PHY_LINK_EXT_EEE_WAKE_ERR_EV
vtss_phy_api.h, 603
vtss_phy_api.h, 603
VTSS_PHY_LINK_EXT_MACSEC_EGRESS_EV
vtss_phy_api.h, 604
VTSS_PHY_LINK_EXT_MACSEC_FC_BUFF_EV
vtss_phy_api.h, 604
VTSS_PHY_LINK_EXT_MACSEC_HOST_MAC_EV
vtss_phy_api.h, 604
VTSS_PHY_LINK_EXT_MACSEC_INGRESS_EV
vtss_phy_api.h, 604
VTSS_PHY_LINK_EXT_MACSEC_LINE_MAC_EV
vtss_phy_api.h, 604
VTSS_PHY_LINK_EXT_MEM_INT_RING_EV
vtss_phy_api.h, 605
VTSS_PHY_LINK_EXT_RR_SW_COMPL_EV
vtss_phy_api.h, 603
VTSS_PHY_LINK_EXTENDED_REG_INT_EV
vtss_phy_api.h, 602
VTSS_PHY_LINK_FALSE_CARRIER_INT_EV
vtss_phy_api.h, 601
VTSS_PHY_LINK_FDX_STATE_CHANGE_EV
vtss_phy_api.h, 600
VTSS_PHY_LINK_FFAIL_EV
vtss_phy_api.h, 599
VTSS_PHY_LINK_INLINE_POW_DEV_DETECT_EV
vtss_phy_api.h, 601
VTSS_PHY_LINK_LINK_SPEED_DS_DETECT_EV
vtss_phy_api.h, 602
VTSS_PHY_LINK_LOS_EV
vtss_phy_api.h, 599
VTSS_PHY_LINK_MASTER_SLAVE_RES_ERR_EV
vtss_phy_api.h, 602
VTSS_PHY_LINK_RX_ER_INT_EV
vtss_phy_api.h, 602
VTSS_PHY_LINK_RX_FIFO_OVERFLOW_INT_EV
vtss_phy_api.h, 601
VTSS_PHY_LINK_SPEED_STATE_CHANGE_EV
vtss_phy_api.h, 600
VTSS_PHY_LINK_SYMBOL_ERR_INT_EV
vtss_phy_api.h, 601
VTSS_PHY_LINK_TX_FIFO_OVERFLOW_INT_EV
vtss_phy_api.h, 601
VTSS_PHY_LINK_UP_FULL_PWR
vtss_phy_api.h, 595
VTSS_PHY_LINK_WAKE_ON_LAN_INT_EV
vtss_phy_api.h, 602
VTSS_PHY_OPT_VERIPH
options.h, 350
VTSS_PHY_PAGE_0x2DAF
vtss_phy_api.h, 598
VTSS_PHY_PAGE_1588
vtss_phy_api.h, 597
VTSS_PHY_PAGE_EXTENDED_2
vtss_phy_api.h, 596
VTSS_PHY_PAGE_EXTENDED_3
vtss_phy_api.h, 597
VTSS_PHY_PAGE_EXTENDED_4
vtss_phy_api.h, 597
VTSS_PHY_PAGE_EXTENDED

vtss_phy_api.h, 596
 VTSS_PHY_PAGE_GPIO
 vtss_phy_api.h, 597
 VTSS_PHY_PAGE_MACSEC
 vtss_phy_api.h, 597
 VTSS_PHY_PAGE_STANDARD
 vtss_phy_api.h, 596
 VTSS_PHY_PAGE_TEST
 vtss_phy_api.h, 598
 VTSS_PHY_PAGE_TR
 vtss_phy_api.h, 598
 VTSS_PHY_POWER_ACTIPHY_BIT
 phy.h, 352
 vtss_phy_api.h, 594
 VTSS_PHY_POWER_DYNAMIC_BIT
 phy.h, 352
 vtss_phy_api.h, 595
 VTSS_PHY_RECov_CLK1
 vtss_phy_api.h, 595
 VTSS_PHY_RECov_CLK2
 vtss_phy_api.h, 596
 VTSS_PHY_RECov_CLK_NUM
 vtss_phy_api.h, 596
 VTSS_PHY_REG_EXTENDED
 vtss_phy_api.h, 598
 VTSS_PHY_REG_GPIO
 vtss_phy_api.h, 599
 VTSS_PHY_REG_STANDARD
 vtss_phy_api.h, 598
 VTSS_PHY_REG_TEST
 vtss_phy_api.h, 599
 VTSS_PHY_REG_TR
 vtss_phy_api.h, 599
 VTSS_PHYS_PORT_CNT
 vtss_fdma_api.h, 416
 VTSS_PORT_ARRAY_SIZE
 types.h, 377
 VTSS_PORT_COUNT
 types.h, 375
 VTSS_PORT_IS_PORT
 types.h, 377
 VTSS_PORT_NO_CPU
 types.h, 376
 VTSS_PORT_NO_END
 types.h, 376
 VTSS_PORT_NO_NONE
 types.h, 376
 VTSS_PORT_NO_START
 types.h, 376
 VTSS_PORT_POLICER_CPU_QUEUES
 vtss_qos_api.h, 676
 VTSS_PORT_POLICERS
 vtss_qos_api.h, 676
 VTSS_PORTS
 types.h, 376
 VTSS_PRIO_ARRAY_SIZE
 types.h, 378
 VTSS_PRIO_END
 types.h, 378
 VTSS_PRIO_NO_NONE
 types.h, 377
 VTSS_PRIO_START
 types.h, 377
 VTSS_PRIO_SUPER
 vtss_packet_api.h, 565
 VTSS_PRIOS
 types.h, 377
 VTSS_PVLAN_ARRAY_SIZE
 vtss_l2_api.h, 459
 VTSS_PVLAN_NO_DEFAULT
 vtss_l2_api.h, 459
 VTSS_PVLAN_NO_END
 vtss_l2_api.h, 459
 VTSS_PVLAN_NO_START
 vtss_l2_api.h, 458
 VTSS_PVLANS
 vtss_l2_api.h, 458
 VTSS_QCE_ID_LAST
 vtss_qos_api.h, 677
 VTSS_QCL_ARRAY_SIZE
 vtss_qos_api.h, 677
 VTSS_QCL_ID_END
 vtss_qos_api.h, 677
 VTSS_QCL_ID_START
 vtss_qos_api.h, 677
 VTSS_QCL_IDS
 vtss_qos_api.h, 677
 VTSS_QUEUE_ARRAY_SIZE
 types.h, 379
 VTSS_QUEUE_END
 types.h, 378
 VTSS_QUEUE_START
 types.h, 378
 VTSS_QUEUES
 types.h, 378
 VTSS_SEC_NS_INTERVAL
 types.h, 391
 VTSS_SVL_PACKET_HDR_SIZE_BYTES
 vtss_packet_api.h, 566
 VTSS_SVL_RX_IFH_SIZE
 vtss_packet_api.h, 567
 VTSS_SYNCE_CLK_MAX
 vtss_sync_api.h, 695
 VTSS_SYNCE_CLK_PORT_ARRAY_SIZE
 types.h, 391
 VTSS_SYNCE_CLK_A
 vtss_sync_api.h, 695
 VTSS_SYNCE_CLK_B
 vtss_sync_api.h, 695
 VTSS_TIME_OF_DAY
 vtss_os_ecos.h, 547
 vtss_os_linux.h, 558
 VTSS_VCE_ID_LAST
 vtss_l2_api.h, 455
 VTSS_VCL_ARRAY_SIZE
 vtss_l2_api.h, 455

VTSS_VCL_ID_END
vtss_l2_api.h, 455

VTSS_VCL_ID_START
vtss_l2_api.h, 455

VTSS_VCL_IDS
vtss_l2_api.h, 455

VTSS_VID_ALL
types.h, 382

VTSS_VID_DEFAULT
types.h, 382

VTSS_VID_NULL
types.h, 381

VTSS_VID_RESERVED
types.h, 382

VTSS_VIDS
types.h, 382

VTSS_VLAN_TRANS_FIRST_GROUP_ID
vtss_l2_api.h, 456

VTSS_VLAN_TRANS_GROUP_MAX_CNT
vtss_l2_api.h, 456

VTSS_VLAN_TRANS_LAST_GROUP_ID
vtss_l2_api.h, 457

VTSS_VLAN_TRANS_MAX_CNT
vtss_l2_api.h, 456

VTSS_VLAN_TRANS_MAX_VLAN_ID
vtss_l2_api.h, 457

VTSS_VLAN_TRANS_NULL_CHECK
vtss_l2_api.h, 458

VTSS_VLAN_TRANS_NULL_GROUP_ID
vtss_l2_api.h, 456

VTSS_VLAN_TRANS_PORT_BF_SIZE
vtss_l2_api.h, 458

VTSS_VLAN_TRANS_VALID_GROUP_CHECK
vtss_l2_api.h, 457

VTSS_VLAN_TRANS_VALID_VLAN_CHECK
vtss_l2_api.h, 457

VTSS_VLAN_TRANS_VID_START
vtss_l2_api.h, 456

val
vtss_eee_port_state_t, 87
vtss_phy_conf_1g_t, 184

value
vtss_sgpio_port_data_t, 284
vtss_vcap_ip_t, 298
vtss_vcap_u128_t, 299
vtss_vcap_u16_t, 300
vtss_vcap_u24_t, 301
vtss_vcap_u32_t, 302
vtss_vcap_u40_t, 303
vtss_vcap_u48_t, 304
vtss_vcap_u8_t, 305
vtss_vcap_vid_t, 307
vtss_vcap_vr_t, 309

vga_adc_debug
vtss_phy_api.h, 634

vid
vtss_ace_vlan_t, 53
vtss_ece_tag_t, 82

vtss_evc_inner_tag_t, 90
vtss_evc_pb_conf_t, 92
vtss_mce_action_t, 135
vtss_mce_key_t, 136
vtss_packet_frame_info_t, 143
vtss_packet_port_info_t, 146
vtss_qce_tag_t, 263
vtss_tci_t, 288
vtss_vce_action_t, 310
vtss_vce_tag_t, 322
vtss_vid_mac_t, 324
vtss_vlan_tag_t, 328
vtss_vlan_trans_grp2vlan_conf_t, 329

vid_mac
vtss_mac_table_entry_t, 131

vid_mode
vtss_evc_inner_tag_t, 90

vlan
vtss_ace_t, 50
vtss_routing_entry_t, 278

vml_format
vtss_debug_info_t, 67

vr
vtss_vcap_vr_t, 310

vtss_ace_add
vtss_security_api.h, 692

vtss_ace_bit_t
vtss_security_api.h, 688

vtss_ace_counter_clear
vtss_security_api.h, 693

vtss_ace_counter_get
vtss_security_api.h, 693

vtss_ace_del
vtss_security_api.h, 692

vtss_ace_frame_arp_t, 31
arp, 31
dip, 33
dmac_match, 32
ethernet, 33
ip, 32
length, 32
req, 31
sip, 33
smac, 31
smac_match, 32
unknown, 32

vtss_ace_frame_etype_t, 33
data, 34
dmac, 34
etype, 34
ptp, 35
smac, 34

vtss_ace_frame_ipv4_t, 35
data, 37
dip, 37
dport, 37
ds, 36
fragment, 36

options, 36
 proto, 36
 ptp, 39
 seq_zero, 39
 sip, 37
 sip_eq_dip, 39
 sip_smac, 40
 sport, 37
 sport_eq_dport, 39
 tcp_ack, 38
 tcp_fin, 38
 tcp_psh, 38
 tcp_rst, 38
 tcp_syn, 38
 tcp_urg, 39
 ttl, 36
vtss_ace_frame_ipv6_t, 40
 data, 41
 dport, 42
 ds, 41
 proto, 41
 ptp, 44
 seq_zero, 44
 sip, 41
 sip_eq_dip, 43
 sport, 42
 sport_eq_dport, 43
 tcp_ack, 43
 tcp_fin, 42
 tcp_psh, 43
 tcp_rst, 42
 tcp_syn, 42
 tcp_urg, 43
 ttl, 41
vtss_ace_frame_llc_t, 44
 dmac, 45
 llc, 45
 smac, 45
vtss_ace_frame_snap_t, 45
 dmac, 46
 smac, 46
 snap, 46
vtss_ace_init
 vtss_security_api.h, 691
vtss_ace_ptp_t, 46
 enable, 47
 header, 47
vtss_ace_sip_smac_t, 47
 enable, 48
 sip, 48
 smac, 48
vtss_ace_t, 48
 action, 50
 arp, 51
 dmac_bc, 50
 dmac_mc, 50
 etype, 51
 frame, 52
 id, 49
 ipv4, 51
 ipv6, 52
 llc, 51
 policy, 49
 port_list, 49
 snap, 51
 type, 50
 vlan, 50
vtss_ace_type_t
 vtss_security_api.h, 687
vtss_ace_vlan_t, 52
 cfi, 53
 tagged, 53
 usr_prio, 53
 vid, 53
vtss_acl_action_t, 54
 cpu, 54
 cpu_once, 54
 cpu_queue, 54
 evc_police, 55
 evc_policer_id, 55
 learn, 55
 mirror, 56
 police, 55
 policer_no, 55
 port_action, 56
 port_list, 56
 ptp_action, 56
vtss_acl_policer_conf_get
 vtss_security_api.h, 689
vtss_acl_policer_conf_set
 vtss_security_api.h, 689
vtss_acl_policer_conf_t, 57
 bit_rate, 57
 bit_rate_enable, 57
 rate, 57
vtss_acl_port_action_t
 vtss_security_api.h, 687
vtss_acl_port_conf_get
 vtss_security_api.h, 690
vtss_acl_port_conf_set
 vtss_security_api.h, 690
vtss_acl_port_conf_t, 58
 action, 58
 policy_no, 58
vtss_acl_port_counter_clear
 vtss_security_api.h, 691
vtss_acl_port_counter_get
 vtss_security_api.h, 691
vtss_acl_ptp_action_t
 vtss_security_api.h, 687
vtss_aggr_mode_get
 vtss_l2_api.h, 487
vtss_aggr_mode_set
 vtss_l2_api.h, 488
vtss_aggr_mode_t, 59
 dmac_enable, 59

sip_dip_enable, 60
 smac_enable, 59
 sport_dport_enable, 60
vtss_aggr_port_members_get
 vtss_l2_api.h, 486
vtss_aggr_port_members_set
 vtss_l2_api.h, 487
vtss_aneg_t, 60
 generate_pause, 61
 obey_pause, 61
vtss_api/include/vtss/api/l2_types.h, 333
vtss_api/include/vtss/api/options.h, 334
vtss_api/include/vtss/api/phy.h, 351
vtss_api/include/vtss/api/port.h, 353
vtss_api/include/vtss/api/types.h, 366
vtss_api/include/vtss_ae_api.h, 403
vtss_api/include/vtss_afi_api.h, 403
vtss_api/include/vtss_aneg_api.h, 403
vtss_api/include/vtss_api.h, 404
vtss_api/include/vtss_evc_api.h, 404
vtss_api/include/vtss_fdma_api.h, 414
vtss_api/include/vtss_gfp_api.h, 432
vtss_api/include/vtss_hqos_api.h, 432
vtss_api/include/vtss_i2c_api.h, 432
vtss_api/include/vtss_init_api.h, 433
vtss_api/include/vtss_l2_api.h, 446
vtss_api/include/vtss_l3_api.h, 504
vtss_api/include/vtss_mac10g_api.h, 504
vtss_api/include/vtss_misc_api.h, 504
vtss_api/include/vtss_mpls_api.h, 539
vtss_api/include/vtss_oam_api.h, 539
vtss_api/include/vtss_oha_api.h, 539
vtss_api/include/vtss_os.h, 540
vtss_api/include/vtss_os_custom.h, 540
vtss_api/include/vtss_os_ecos.h, 545
vtss_api/include/vtss_os_linux.h, 555
vtss_api/include/vtss_otn_api.h, 562
vtss_api/include/vtss_packet_api.h, 562
vtss_api/include/vtss_pcs_10gbase_r_api.h, 584
vtss_api/include/vtss_phy_10g_api.h, 584
vtss_api/include/vtss_phy_api.h, 584
vtss_api/include/vtss_phy_ts_api.h, 661
vtss_api/include/vtss_port_api.h, 662
vtss_api/include/vtss_qos_api.h, 674
vtss_api/include/vtss_rab_api.h, 683
vtss_api/include/vtss_security_api.h, 683
vtss_api/include/vtss_sf14_api.h, 693
vtss_api/include/vtss_sync_api.h, 694
vtss_api/include/vtss_tfi5_api.h, 697
vtss_api/include/vtss_ts_api.h, 698
vtss_api/include/vtss_upi_api.h, 714
vtss_api/include/vtss_wis_api.h, 714
vtss_api/include/vtss_xaui_api.h, 714
vtss_api/include/vtss_xfi_api.h, 715
vtss_api_lock_t, 61
 file, 62
 function, 62
 inst, 62
 line, 62
vtss_apvlan_port_members_get
 vtss_l2_api.h, 483
vtss_apvlan_port_members_set
 vtss_l2_api.h, 484
vtss_auth_port_state_get
 vtss_security_api.h, 688
vtss_auth_port_state_set
 vtss_security_api.h, 688
vtss_auth_state_t
 vtss_security_api.h, 686
vtss_basic_counters_t, 63
 rx_frames, 63
 tx_frames, 63
vtss_callout_free
 vtss_os_ecos.h, 554
vtss_callout_lock
 vtss_misc_api.h, 520
vtss_callout_malloc
 vtss_os_ecos.h, 554
vtss_callout_trace_hex_dump
 vtss_misc_api.h, 518
vtss_callout_trace_printf
 vtss_misc_api.h, 518
vtss_callout_unlock
 vtss_misc_api.h, 520
vtss_chip_id_get
 vtss_misc_api.h, 523
vtss_chip_id_t, 63
 part_number, 64
 revision, 64
vtss_counter_pair_t, 64
 bytes, 65
 frames, 65
vtss_debug_group_t
 vtss_misc_api.h, 514
vtss_debug_info_get
 vtss_misc_api.h, 519
vtss_debug_info_print
 vtss_misc_api.h, 519
vtss_debug_info_t, 65
 chip_no, 66
 clear, 67
 full, 66
 group, 66
 layer, 66
 port_list, 66
 vml_format, 67
vtss_debug_layer_t
 vtss_misc_api.h, 512
vtss_debug_lock
 vtss_misc_api.h, 520
vtss_debug_lock_t, 67
 chip_no, 68
vtss_debug_reg_check_set
 vtss_misc_api.h, 538
vtss_debug_unlock
 vtss_misc_api.h, 521

vtss_dev_all_event_enable
 vtss_misc_api.h, 525
 vtss_dev_all_event_poll
 vtss_misc_api.h, 524
 vtss_dgroup_port_conf_get
 vtss_l2_api.h, 484
 vtss_dgroup_port_conf_set
 vtss_l2_api.h, 484
 vtss_dgroup_port_conf_t, 68
 dgroup_no, 68
 vtss_dlb_policer_conf_t, 69
 cbs, 70
 cf, 69
 cir, 70
 ebs, 70
 eir, 70
 enable, 69
 line_rate, 70
 type, 69
 vtss_dscp_emode_t
 vtss_qos_api.h, 678
 vtss_dscp_mode_t
 vtss_qos_api.h, 678
 vtss_ece_action_t, 71
 dir, 71
 evc_id, 72
 outer_tag, 72
 policy_no, 72
 pop_tag, 71
 prio, 72
 prio_enable, 72
 vtss_ece_add
 vtss_evc_api.h, 412
 vtss_ece_del
 vtss_evc_api.h, 413
 vtss_ece_dir_t
 types.h, 401
 vtss_ece_frame_ipv4_t, 73
 dport, 74
 dscp, 73
 fragment, 73
 proto, 74
 sip, 74
 sport, 74
 vtss_ece_frame_ipv6_t, 75
 dport, 76
 dscp, 75
 proto, 75
 sip, 75
 sport, 76
 vtss_ece_init
 vtss_evc_api.h, 412
 vtss_ece_key_t, 76
 frame, 78
 ipv4, 77
 ipv6, 78
 mac, 77
 port_list, 77
 tag, 77
 type, 77
 vtss_ece_mac_t, 78
 dmac_bc, 79
 dmac_mc, 79
 smac, 79
 vtss_ece_outer_tag_t, 79
 dei, 80
 enable, 80
 pcp, 80
 pcp_dei_preserve, 80
 vtss_ece_pop_tag_t
 types.h, 402
 vtss_ece_port_t
 vtss_evc_api.h, 408
 vtss_ece_t, 81
 action, 81
 id, 81
 key, 81
 vtss_ece_tag_t, 82
 dei, 83
 pcp, 82
 s_tagged, 83
 tagged, 83
 vid, 82
 vtss_ece_type_t
 vtss_evc_api.h, 408
 vtss_eee_port_conf_set
 vtss_misc_api.h, 537
 vtss_eee_port_conf_t, 83
 eee_ena, 84
 eee_fast_queues, 84
 lp_advertisement, 84
 optimized_for_power, 85
 tx_tw, 84
 vtss_eee_port_counter_get
 vtss_misc_api.h, 538
 vtss_eee_port_counter_t, 85
 fill_level, 86
 fill_level_get, 85
 fill_level_thres, 86
 tx_out_bytes, 86
 tx_out_bytes_get, 86
 vtss_eee_port_state_set
 vtss_misc_api.h, 537
 vtss_eee_port_state_t, 87
 select, 87
 val, 87
 vtss_eee_state_select_t
 vtss_misc_api.h, 516
 vtss_eps_port_conf_get
 vtss_l2_api.h, 499
 vtss_eps_port_conf_set
 vtss_l2_api.h, 500
 vtss_eps_port_conf_t, 87
 port_no, 88
 type, 88
 vtss_eps_port_selector_get

vtss_l2_api.h, 500
vtss_eps_port_selector_set
 vtss_l2_api.h, 500
vtss_eps_port_type_t
 vtss_l2_api.h, 462
vtss_eps_selector_t
 vtss_l2_api.h, 462
vtss_erps_port_state_get
 vtss_l2_api.h, 503
vtss_erps_port_state_set
 vtss_l2_api.h, 503
vtss_erps_state_t
 vtss_l2_api.h, 462
vtss_erps_vlan_member_get
 vtss_l2_api.h, 501
vtss_erps_vlan_member_set
 vtss_l2_api.h, 501
vtss_evc_add
 vtss_evc_api.h, 410
vtss_evc_api.h
 VTSS_ECE_ID_LAST, 407
 VTSS_EVC_ID_NONE, 406
 VTSS_EVC_POLICERS, 406
 VTSS_MCE_ID_LAST, 407
 VTSS_MCE_POP_NONE, 407
vtss_ece_add, 412
vtss_ece_del, 413
vtss_ece_init, 412
vtss_ece_port_t, 408
vtss_ece_type_t, 408
vtss_evc_add, 410
vtss_evc_del, 411
vtss_evc_get, 411
vtss_evc_inner_tag_type_t, 408
vtss_evc_policer_conf_get, 410
vtss_evc_policer_conf_set, 410
vtss_evc_port_conf_get, 409
vtss_evc_port_conf_set, 409
vtss_evc_vid_mode_t, 407
vtss_mce_add, 413
vtss_mce_del, 414
vtss_mce_init, 413
vtss_evc_conf_t, 88
 learning, 89
 network, 89
 pb, 89
vtss_evc_del
 vtss_evc_api.h, 411
vtss_evc_get
 vtss_evc_api.h, 411
vtss_evc_inner_tag_t, 90
 dei, 91
 pcp, 91
 pcp_dei_preserve, 91
 type, 90
 vid, 90
 vid_mode, 90
vtss_evc_inner_tag_type_t

vtss_evc_api.h, 408
vtss_evc_pb_conf_t, 91
 inner_tag, 93
 ivid, 92
 nni, 92
 uvid, 92
 vid, 92
vtss_evc_policer_conf_get
 vtss_evc_api.h, 410
vtss_evc_policer_conf_set
 vtss_evc_api.h, 410
vtss_evc_port_conf_get
 vtss_evc_api.h, 409
vtss_evc_port_conf_set
 vtss_evc_api.h, 409
vtss_evc_port_conf_t, 93
 dei_colouring, 93
 dmac_dip, 94
 inner_tag, 94
vtss_evc_vid_mode_t
 vtss_evc_api.h, 407
vtss_fan_conf_t, 94
 fan_low_pol, 95
 fan_open_col, 95
 fan_pwm_freq, 95
 ppr, 95
 type, 95
vtss_fan_controller_init
 vtss_misc_api.h, 536
vtss_fan_cool_lvl_get
 vtss_misc_api.h, 536
vtss_fan_cool_lvl_set
 vtss_misc_api.h, 536
vtss_fan_rotation_get
 vtss_misc_api.h, 535
vtss_fdma_afi_cancel
 vtss_fdma_api.h, 427
vtss_fdma_afi_frm_cnt
 vtss_fdma_api.h, 427
vtss_fdma_afi_type_t
 vtss_fdma_api.h, 422
vtss_fdma_api.h
 VTSS_AFI_FPS_MAX, 419
 VTSS_FDMA_CCM_FPS_MAX, 419
 VTSS_FDMA_CCM_FREQ_LIST_LEN, 419
 VTSS_FDMA_CCM_QUOTIENT_MAX, 419
 VTSS_FDMA_CH_CNT, 416
 VTSS_FDMA_DCB_SIZE_BYTES, 417
 VTSS_FDMA_HDR_SIZE_BYTES, 417
 VTSS_FDMA_MAX_DATA_PER_DCB_BYTES,
 417
 VTSS_FDMA_MAX_FRAME_SIZE_BYTES, 418
 VTSS_FDMA_MIN_DATA_PER_INJ_SOF_DC←
 B_BYTES, 417
 VTSS_FDMA_MIN_DATA_PER_NON_SOF_D←
 CB_BYTES, 418
 VTSS_FDMA_MIN_DATA_PER_XTR_SOF_DC←
 B_BYTES, 418

VTSS_FDMA_MIN_FRAME_SIZE_BYTES, 417
 VTSS_OS_COMPILER_ATTRIBUTE_ALIGNED, 418
 VTSS_OS_DCACHE_LINE_SIZE_BYTES, 418
 VTSS_PHYS_PORT_CNT, 416
 vtss_fdma_afi_cancel, 427
 vtss_fdma_afi_frm_cnt, 427
 vtss_fdma_afi_type_t, 422
 vtss_fdma_cfg, 424
 vtss_fdma_ch_t, 420
 vtss_fdma_ch_usage_t, 423
 vtss_fdma_dcb_get, 428
 vtss_fdma_dcb_release, 425
 vtss_fdma_dcb_type_t, 423
 vtss_fdma_irq_handler, 431
 vtss_fdma_list_t, 422
 vtss_fdma_stats_clr, 430
 vtss_fdma_throttle_cfg_get, 428
 vtss_fdma_throttle_cfg_set, 429
 vtss_fdma_throttle_tick, 430
 vtss_fdma_tx, 425
 vtss_fdma_tx_info_init, 426
 vtss_fdma_uninit, 423
 vtss_fdma_cfg
 vtss_fdma_api.h, 424
 vtss_fdma_cfg_t, 96
 afi_buf_cnt, 100
 afi_done_cb, 101
 enable, 96
 rx_alloc_cb, 97
 rx_allow_multiple_dcbs, 99
 rx_allow_vlan_tag_mismatch, 99
 rx_buf_cnt, 97
 rx_cb, 99
 rx_dont_reinsert_vlan_tag, 99
 rx_dont_strip_vlan_tag, 98
 rx_mtu, 97
 tx_buf_cnt, 100
 tx_done_cb, 100
 vtss_fdma_ch_cfg_t, 101
 ccm_quotient_max, 105
 chip_no, 104
 inj_grp_mask, 102
 list, 103
 prio, 104
 usage, 102
 xtr_cb, 103
 xtr_grp, 102
 vtss_fdma_ch_t
 vtss_fdma_api.h, 420
 vtss_fdma_ch_usage_t
 vtss_fdma_api.h, 423
 vtss_fdma_dcb_get
 vtss_fdma_api.h, 428
 vtss_fdma_dcb_release
 vtss_fdma_api.h, 425
 vtss_fdma_dcb_type_t
 vtss_fdma_api.h, 423
 vtss_fdma_irq_handler
 vtss_fdma_api.h, 431
 vtss_fdma_list_t
 vtss_fdma_api.h, 422
 vtss_fdma_stats_clr
 vtss_fdma_api.h, 430
 vtss_fdma_throttle_cfg_get
 vtss_fdma_api.h, 428
 vtss_fdma_throttle_cfg_set
 vtss_fdma_api.h, 429
 vtss_fdma_throttle_cfg_t, 105
 byte_limit_per_tick, 106
 frm_limit_per_tick, 106
 suspend_tick_cnt, 107
 vtss_fdma_throttle_tick
 vtss_fdma_api.h, 430
 vtss_fdma_tx
 vtss_fdma_api.h, 425
 vtss_fdma_tx_info_init
 vtss_fdma_api.h, 426
 vtss_fdma_tx_info_t, 107
 afi_enable_counting, 109
 afi_enable_sequence_numbering, 109
 afi_fps, 108
 afi_sequence_number_offset, 110
 afi_type, 109
 pre_cb, 108
 pre_cb_ctxt1, 108
 pre_cb_ctxt2, 108
 vtss_fdma_uninit
 vtss_fdma_api.h, 423
 vtss_fefi_mode_t
 vtss_phy_api.h, 615
 vtss_fiber_port_speed_t
 port.h, 365
 vtss_gpio_direction_set
 vtss_misc_api.h, 526
 vtss_gpio_event_enable
 vtss_misc_api.h, 528
 vtss_gpio_event_poll
 vtss_misc_api.h, 527
 vtss_gpio_mode_set
 vtss_misc_api.h, 525
 vtss_gpio_mode_t
 vtss_misc_api.h, 515
 vtss_gpio_read
 vtss_misc_api.h, 526
 vtss_gpio_write
 vtss_misc_api.h, 527
 vtss_hqos_sch_mode_t
 types.h, 402
 vtss_i2c_read_t
 vtss_init_api.h, 436
 vtss_i2c_write_t
 vtss_init_api.h, 437
 vtss_init_api.h
 VTSS_I2C_NO_MULTIPLEXER, 435
 vtss_i2c_read_t, 436

vtss_i2c_write_t, 437
vtss_init_conf_get, 444
vtss_init_conf_set, 444
vtss_inst_create, 443
vtss_inst_destroy, 444
vtss_inst_get, 443
vtss_miim_read_t, 439
vtss_miim_write_t, 439
vtss_mmd_read_inc_t, 440
vtss_mmd_read_t, 439
vtss_mmd_write_t, 440
vtss_port_mux_mode_t, 442
vtss_reg_read_t, 435
vtss_reg_write_t, 436
vtss_restart_conf_end, 445
vtss_restart_conf_get, 445
vtss_restart_conf_set, 446
vtss_restart_status_get, 445
vtss_restart_t, 442
vtss_spi_32bit_read_write_t, 438
vtss_spi_64bit_read_write_t, 438
vtss_spi_read_write_t, 437
vtss_target_type_t, 441
vtss_init_conf_get
 vtss_init_api.h, 444
vtss_init_conf_set
 vtss_init_api.h, 444
vtss_init_conf_t, 111
 miim_read, 112
 miim_write, 112
 mmd_read, 112
 mmd_read_inc, 112
 mmd_write, 112
 mux_mode, 114
 pi, 114
 reg_read, 111
 reg_write, 111
 restart_info_port, 114
 restart_info_src, 113
 serdes, 114
 spi_32bit_read_write, 113
 spi_64bit_read_write, 113
 spi_read_write, 113
 warm_start_enable, 113
vtss_inst_create
 vtss_init_api.h, 443
vtss_inst_create_t, 115
 target, 115
vtss_inst_destroy
 vtss_init_api.h, 444
vtss_inst_get
 vtss_init_api.h, 443
vtss_internal_bw_t
 vtss_port_api.h, 665
vtss_intr_cfg
 vtss_misc_api.h, 530
vtss_intr_mask_set
 vtss_misc_api.h, 531
vtss_intr_pol_negation
 vtss_misc_api.h, 532
vtss_intr_status_get
 vtss_misc_api.h, 531
vtss_intr_sticky_clear
 vtss_misc_api.h, 522
vtss_intr_t, 115
 link_change, 116
vtss_ip_addr_t, 116
 addr, 117
 ipv4, 117
 ipv6, 117
 type, 116
vtss_ip_network_t, 117
 address, 118
 prefix_size, 118
vtss_ip_type_t
 types.h, 400
vtss_ipv4_mc_add
 vtss_l2_api.h, 495
vtss_ipv4_mc_del
 vtss_l2_api.h, 495
vtss_ipv4_mc_flood_members_get
 vtss_l2_api.h, 494
vtss_ipv4_mc_flood_members_set
 vtss_l2_api.h, 494
vtss_ipv4_network_t, 118
 address, 119
 prefix_size, 119
vtss_ipv4_uc_t, 119
 destination, 120
 network, 120
vtss_ipv6_mc_add
 vtss_l2_api.h, 498
vtss_ipv6_mc_ctrl_flood_get
 vtss_l2_api.h, 498
vtss_ipv6_mc_ctrl_flood_set
 vtss_l2_api.h, 498
vtss_ipv6_mc_del
 vtss_l2_api.h, 499
vtss_ipv6_mc_flood_members_get
 vtss_l2_api.h, 497
vtss_ipv6_mc_flood_members_set
 vtss_l2_api.h, 497
vtss_ipv6_network_t, 120
 address, 121
 prefix_size, 121
vtss_ipv6_t, 121
 addr, 122
vtss_ipv6_uc_t, 122
 destination, 122
 network, 122
vtss_irq_conf_get
 vtss_misc_api.h, 532
vtss_irq_conf_set
 vtss_misc_api.h, 532
vtss_irq_conf_t, 123
 destination, 123

external, 123
vtss_irq_enable
 vtss_misc_api.h, 533
vtss_irq_status_get_and_mask
 vtss_misc_api.h, 533
vtss_irq_status_t, 124
 active, 124
 raw_ident, 124
 raw_mask, 125
 raw_status, 125
vtss_irq_t
 vtss_misc_api.h, 516
vtss_isidx_t
 types.h, 394
vtss_isolated_port_members_get
 vtss_l2_api.h, 481
vtss_isolated_port_members_set
 vtss_l2_api.h, 482
vtss_isolated_vlan_get
 vtss_l2_api.h, 481
vtss_isolated_vlan_set
 vtss_l2_api.h, 481
vtss_l2_api.h
 VTSS_ERPI_ARRAY_SIZE, 460
 VTSS_ERPI_END, 460
 VTSS_ERPI_START, 459
 VTSS_ERPIS, 459
 VTSS_MAC_ADDRS, 454
 VTSS_MSTI_ARRAY_SIZE, 454
 VTSS_MSTI_END, 454
 VTSS_MSTI_START, 454
 VTSS_MSTIS, 454
 VTSS_PVLAN_ARRAY_SIZE, 459
 VTSS_PVLAN_NO_DEFAULT, 459
 VTSS_PVLAN_NO_END, 459
 VTSS_PVLAN_NO_START, 458
 VTSS_PVLANS, 458
 VTSS_VCE_ID_LAST, 455
 VTSS_VCL_ARRAY_SIZE, 455
 VTSS_VCL_ID_END, 455
 VTSS_VCL_ID_START, 455
 VTSS_VCL_IDS, 455
 VTSS_VLAN_TRANS_FIRST_GROUP_ID, 456
 VTSS_VLAN_TRANS_GROUP_MAX_CNT, 456
 VTSS_VLAN_TRANS_LAST_GROUP_ID, 457
 VTSS_VLAN_TRANS_MAX_CNT, 456
 VTSS_VLAN_TRANS_MAX_VLAN_ID, 457
 VTSS_VLAN_TRANS_NULL_CHECK, 458
 VTSS_VLAN_TRANS_NULL_GROUP_ID, 456
 VTSS_VLAN_TRANS_PORT_BF_SIZE, 458
 VTSS_VLAN_TRANS_VALID_GROUP_CHECK,
 457
 VTSS_VLAN_TRANS_VALID_VLAN_CHECK,
 457
 VTSS_VLAN_TRANS_VID_START, 456
vtss_aggr_mode_get, 487
vtss_aggr_mode_set, 488
vtss_aggr_port_members_get, 486
vtss_aggr_port_members_set, 487
vtss_apvlan_port_members_get, 483
vtss_apvlan_port_members_set, 484
vtss_dgroup_port_conf_get, 484
vtss_dgroup_port_conf_set, 484
vtss_eps_port_conf_get, 499
vtss_eps_port_conf_set, 500
vtss_eps_port_selector_get, 500
vtss_eps_port_selector_set, 500
vtss_eps_port_type_t, 462
vtss_eps_selector_t, 462
vtss_erps_port_state_get, 503
vtss_erps_port_state_set, 503
vtss_erps_state_t, 462
vtss_erps_vlan_member_get, 501
vtss_erps_vlan_member_set, 501
vtss_ipv4_mc_add, 495
vtss_ipv4_mc_del, 495
vtss_ipv4_mc_flood_members_get, 494
vtss_ipv4_mc_flood_members_set, 494
vtss_ipv6_mc_add, 498
vtss_ipv6_mc_ctrl_flood_get, 498
vtss_ipv6_mc_ctrl_flood_set, 498
vtss_ipv6_mc_del, 499
vtss_ipv6_mc_flood_members_get, 497
vtss_ipv6_mc_flood_members_set, 497
vtss_isolated_port_members_get, 481
vtss_isolated_port_members_set, 482
vtss_isolated_vlan_get, 481
vtss_isolated_vlan_set, 481
vtss_learn_port_mode_get, 468
vtss_learn_port_mode_set, 468
vtss_mac_table_add, 463
vtss_mac_table_age, 465
vtss_mac_table_age_time_get, 464
vtss_mac_table_age_time_set, 465
vtss_mac_table_del, 463
vtss_mac_table_flush, 466
vtss_mac_table_get, 463
vtss_mac_table_get_next, 464
vtss_mac_table_port_flush, 466
vtss_mac_table_status_get, 467
vtss_mac_table_vlan_age, 465
vtss_mac_table_vlan_flush, 467
vtss_mac_table_vlan_port_flush, 467
vtss_mc_flood_members_get, 493
vtss_mc_flood_members_set, 494
vtss_mirror_conf_get, 488
vtss_mirror_conf_set, 488
vtss_mirror_cpu_egress_get, 492
vtss_mirror_cpu_egress_set, 492
vtss_mirror_cpu_ingress_get, 491
vtss_mirror_cpu_ingress_set, 491
vtss_mirror_egress_ports_get, 490
vtss_mirror_egress_ports_set, 491
vtss_mirror_ingress_ports_get, 489
vtss_mirror_ingress_ports_set, 490
vtss_mirror_monitor_port_get, 489

vtss_mirror_monitor_port_set, 489
vtss_mstp_port_msti_state_get, 471
vtss_mstp_port_msti_state_set, 472
vtss_mstp_vlan_msti_get, 470
vtss_mstp_vlan_msti_set, 471
vtss_port_state_get, 469
vtss_port_state_set, 469
vtss_pvlan_port_members_get, 482
vtss_pvlan_port_members_set, 483
vtss_sfflow_port_conf_get, 485
vtss_sfflow_port_conf_set, 485
vtss_sfflow_sampling_rate_convert, 486
vtss_stp_port_state_get, 469
vtss_stp_port_state_set, 470
vtss_stp_state_t, 460
vtss_uc_flood_members_get, 493
vtss_uc_flood_members_set, 493
vtss_vce_add, 478
vtss_vce_del, 478
vtss_vce_init, 477
vtss_vce_type_t, 461
vtss_vcl_port_conf_get, 476
vtss_vcl_port_conf_set, 477
vtss_vlan_conf_get, 472
vtss_vlan_conf_set, 472
vtss_vlan_port_conf_get, 473
vtss_vlan_port_conf_set, 473
vtss_vlan_port_members_get, 474
vtss_vlan_port_members_set, 474
vtss_vlan_port_type_t, 461
vtss_vlan_trans_group_add, 478
vtss_vlan_trans_group_del, 479
vtss_vlan_trans_group_get, 479
vtss_vlan_trans_group_to_port_get, 480
vtss_vlan_trans_group_to_port_set, 480
vtss_vlan_tx_tag_get, 475
vtss_vlan_tx_tag_set, 476
vtss_vlan_tx_tag_t, 461
vtss_vlan_vid_conf_get, 475
vtss_vlan_vid_conf_set, 475
vtss_vt_id_t, 460
vtss_l3_counters_t, 125
 ipv4uc_received_frames, 126
 ipv4uc_received_octets, 126
 ipv4uc_transmitted_frames, 127
 ipv4uc_transmitted_octets, 126
 ipv6uc_received_frames, 126
 ipv6uc_received_octets, 126
 ipv6uc_transmitted_frames, 127
 ipv6uc_transmitted_octets, 127
vtss_lcpll_status_t, 127
 cal_done, 128
 cal_error, 128
 fsm_lock, 128
 fsm_stat, 129
 gain_stat, 129
 lock_status, 128
vtss_learn_mode_t, 129
 automatic, 130
 cpu, 130
 discard, 130
vtss_learn_port_mode_get
 vtss_l2_api.h, 468
vtss_learn_port_mode_set
 vtss_l2_api.h, 468
vtss_mac_addr_t
 types.h, 394
vtss_mac_t, 130
 addr, 131
vtss_mac_table_add
 vtss_l2_api.h, 463
vtss_mac_table_age
 vtss_l2_api.h, 465
vtss_mac_table_age_time_get
 vtss_l2_api.h, 464
vtss_mac_table_age_time_set
 vtss_l2_api.h, 465
vtss_mac_table_del
 vtss_l2_api.h, 463
vtss_mac_table_entry_t, 131
 aged, 132
 copy_to_cpu, 132
 cpu_queue, 132
 destination, 132
 locked, 132
 vid_mac, 131
vtss_mac_table_flush
 vtss_l2_api.h, 466
vtss_mac_table_get
 vtss_l2_api.h, 463
vtss_mac_table_get_next
 vtss_l2_api.h, 464
vtss_mac_table_port_flush
 vtss_l2_api.h, 466
vtss_mac_table_status_get
 vtss_l2_api.h, 467
vtss_mac_table_status_t, 133
 aged, 134
 learned, 133
 moved, 134
 replaced, 133
vtss_mac_table_vlan_age
 vtss_l2_api.h, 465
vtss_mac_table_vlan_flush
 vtss_l2_api.h, 467
vtss_mac_table_vlan_port_flush
 vtss_l2_api.h, 467
vtss_mc_flood_members_get
 vtss_l2_api.h, 493
vtss_mc_flood_members_set
 vtss_l2_api.h, 494
vtss_mce_action_t, 134
 policy_no, 135
 pop_cnt, 135
 prio, 135
 prio_enable, 135

vid, 135
 vtss_mce_add
 vtss_evc_api.h, 413
 vtss_mce_del
 vtss_evc_api.h, 414
 vtss_mce_init
 vtss_evc_api.h, 413
 vtss_mce_key_t, 136
 data, 137
 port_list, 136
 vid, 136
 vtss_mce_t, 137
 action, 138
 id, 138
 key, 138
 tt_loop, 137
 vtss_mem_flags_t
 types.h, 397
 vtss_mep_policer_conf_get
 vtss_qos_api.h, 680
 vtss_mep_policer_conf_set
 vtss_qos_api.h, 680
 vtss_miim_controller_t
 vtss_port_api.h, 665
 vtss_miim_read
 vtss_port_api.h, 673
 vtss_miim_read_t
 vtss_init_api.h, 439
 vtss_miim_write
 vtss_port_api.h, 673
 vtss_miim_write_t
 vtss_init_api.h, 439
 vtss_mirror_conf_get
 vtss_l2_api.h, 488
 vtss_mirror_conf_set
 vtss_l2_api.h, 488
 vtss_mirror_conf_t, 138
 fwd_enable, 139
 port_no, 139
 vtss_mirror_cpu_egress_get
 vtss_l2_api.h, 492
 vtss_mirror_cpu_egress_set
 vtss_l2_api.h, 492
 vtss_mirror_cpu_ingress_get
 vtss_l2_api.h, 491
 vtss_mirror_cpu_ingress_set
 vtss_l2_api.h, 491
 vtss_mirror_egress_ports_get
 vtss_l2_api.h, 490
 vtss_mirror_egress_ports_set
 vtss_l2_api.h, 491
 vtss_mirror_ingress_ports_get
 vtss_l2_api.h, 489
 vtss_mirror_ingress_ports_set
 vtss_l2_api.h, 490
 vtss_mirror_monitor_port_get
 vtss_l2_api.h, 489
 vtss_mirror_monitor_port_set

 vtss_l2_api.h, 489
 vtss_misc_api.h
 tod_get_ns_cnt_cb_t, 511
 VTSS_OS_TIMESTAMP_TYPE, 510
 VTSS_OS_TIMESTAMP, 510
 vtss_callout_lock, 520
 vtss_callout_trace_hex_dump, 518
 vtss_callout_trace_printf, 518
 vtss_callout_unlock, 520
 vtss_chip_id_get, 523
 vtss_debug_group_t, 514
 vtss_debug_info_get, 519
 vtss_debug_info_print, 519
 vtss_debug_layer_t, 512
 vtss_debug_lock, 520
 vtss_debug_reg_check_set, 538
 vtss_debug_unlock, 521
 vtss_dev_all_event_enable, 525
 vtss_dev_all_event_poll, 524
 vtss_eee_port_conf_set, 537
 vtss_eee_port_counter_get, 538
 vtss_eee_port_state_set, 537
 vtss_eee_state_select_t, 516
 vtss_fan_controller_init, 536
 vtss_fan_cool_lvl_get, 536
 vtss_fan_cool_lvl_set, 536
 vtss_fan_rotation_get, 535
 vtss_gpio_direction_set, 526
 vtss_gpio_event_enable, 528
 vtss_gpio_event_poll, 527
 vtss_gpio_mode_set, 525
 vtss_gpio_mode_t, 515
 vtss_gpio_read, 526
 vtss_gpio_write, 527
 vtss_intr_cfg, 530
 vtss_intr_mask_set, 531
 vtss_intr_pol_negation, 532
 vtss_intr_status_get, 531
 vtss_intr_sticky_clear, 522
 vtss_irq_conf_get, 532
 vtss_irq_conf_set, 532
 vtss_irq_enable, 533
 vtss_irq_status_get_and_mask, 533
 vtss_irq_t, 516
 vtss_poll_1sec, 523
 vtss_ptp_event_enable, 524
 vtss_ptp_event_poll, 523
 vtss_reg_read, 521
 vtss_reg_write, 521
 vtss_reg_write_masked, 522
 vtss_sgpiobmode_t, 516
 vtss_sgpiocfg_get, 528
 vtss_sgpiocfg_set, 528
 vtss_sgpio_event_enable, 530
 vtss_sgpio_event_poll, 529
 vtss_sgpiomode_t, 515
 vtss_sgpioread, 529
 vtss_temp_sensor_get, 535

vtss_temp_sensor_init, 534
vtss_tod_get_ns_cnt, 534
vtss_tod_set_ns_cnt_cb, 534
vtss_trace_conf_get, 517
vtss_trace_conf_set, 517
vtss_trace_group_t, 511
vtss_trace_layer_t, 511
vtss_trace_level_t, 512
vtss_mmd_read_inc_t
 vtss_init_api.h, 440
vtss_mmd_read_t
 vtss_init_api.h, 439
vtss_mmd_write_t
 vtss_init_api.h, 440
vtss_mstp_port_msti_state_get
 vtss_l2_api.h, 471
vtss_mstp_port_msti_state_set
 vtss_l2_api.h, 472
vtss_mstp_vlan_msti_get
 vtss_l2_api.h, 470
vtss_mstp_vlan_msti_set
 vtss_l2_api.h, 471
vtss_mtimmer_t, 139
 now, 140
 timeout, 140
 vtss_os_custom.h, 544
 vtss_os_ecos.h, 553
vtss_npi_conf_get
 vtss_packet_api.h, 571
vtss_npi_conf_set
 vtss_packet_api.h, 571
vtss_npi_conf_t, 140
 enable, 141
 port_no, 141
vtss_os_custom.h
 uint, 541
 ulong, 541
 VTSS_DIV64, 542
 VTSS_LABS, 542
 VTSS_LLabs, 543
 VTSS_MOD64, 542
 VTSS_MSLEEP, 541
 VTSS_MTIMER_CANCEL, 542
 VTSS_MTIMER_START, 541
 VTSS_MTIMER_TIMEOUT, 541
 VTSS_OS_CTZ64, 543
 VTSS_OS_CTZ, 543
 VTSS_OS_FREE, 544
 VTSS_OS_MALLOC, 543
 VTSS_OS_RAND, 544
 vtss_mtimmer_t, 544
vtss_os_ecos.h
 llabs, 553
 VTSS_DIV64, 547
 VTSS_LABS, 548
 VTSS_LLabs, 548
 VTSS_MOD64, 547
 VTSS_MSLEEP, 546
VTSS_MTIMER_CANCEL, 547
VTSS_MTIMER_START, 546
VTSS_MTIMER_TIMEOUT, 547
VTSS_NSLEEP, 546
VTSS_OS_BIG_ENDIAN, 551
VTSS_OS_COMPILER_ATTRIBUTE_ALIGNED, 550
VTSS_OS_CTZ64, 548
VTSS_OS_CTZ, 548
VTSS_OS_DCACHE_FLUSH, 551
VTSS_OS_DCACHE_INVALIDATE, 550
VTSS_OS_DCACHE_LINE_SIZE_BYTES, 550
VTSS_OS_FREE, 549
VTSS_OS_INTERRUPT_DISABLE, 553
VTSS_OS_INTERRUPT_FLAGS, 552
VTSS_OS_INTERRUPT_RESTORE, 553
VTSS_OS_MALLOC, 549
VTSS_OS_NTOHL, 551
VTSS_OS_RAND, 549
VTSS_OS_REORDER_BARRIER, 550
VTSS_OS_SCHEDULER_FLAGS, 552
VTSS_OS_SCHEDULER_LOCK, 552
VTSS_OS_SCHEDULER_UNLOCK, 552
VTSS_OS_VIRT_TO_PHYS, 551
VTSS_TIME_OF_DAY, 547
vtss_callout_free, 554
vtss_callout_malloc, 554
vtss_mtimmer_t, 553
vtss_os_linux.h
 VTSS_DIV64, 559
 VTSS_LABS, 559
 VTSS_LLabs, 559
 VTSS_MOD64, 559
 VTSS_MSLEEP, 556
 VTSS_MTIMER_CANCEL, 557
 VTSS_MTIMER_START, 557
 VTSS_MTIMER_TIMEOUT, 557
 VTSS_NSLEEP, 556
 VTSS_OS_BIG_ENDIAN, 556
 VTSS_OS_CTZ64, 561
 VTSS_OS_CTZ, 560
 VTSS_OS_FREE, 561
 VTSS_OS_MALLOC, 561
 VTSS_OS_NTOHL, 556
 VTSS_OS_RAND, 562
 VTSS_OS_SCHEDULER_FLAGS, 558
 VTSS_OS_SCHEDULER_LOCK, 558
 VTSS_OS_SCHEDULER_UNLOCK, 558
 VTSS_TIME_OF_DAY, 558
vtss_os_timestamp_t, 141
 hw_cnt, 142
vtss_packet_api.h
 VTSS_JR1_PACKET_HDR_SIZE_BYTES, 566
 VTSS_JR1_RX_IFH_SIZE, 567
 VTSS_JR2_PACKET_HDR_SIZE_BYTES, 566
 VTSS_JR2_RX_IFH_SIZE, 567
 VTSS_L26_PACKET_HDR_SIZE_BYTES, 566
 VTSS_L26_RX_IFH_SIZE, 567

VTSS_PACKET_HDR_SIZE_BYTES, 566
 VTSS_PACKET_TX_IFH_MAX, 567
 VTSS_PRIO_SUPER, 565
 VTSS_SVL_PACKET_HDR_SIZE_BYTES, 566
 VTSS_SVL_RX_IFH_SIZE, 567
 vtss_npi_conf_get, 571
 vtss_npi_conf_set, 571
 vtss_packet_dma_conf_get, 582
 vtss_packet_dma_conf_set, 583
 vtss_packet_dma_offset, 583
 vtss_packet_filter_t, 568
 vtss_packet_frame_filter, 577
 vtss_packet_frame_info_init, 577
 vtss_packet_oam_type_t, 568
 vtss_packet_port_filter_get, 578
 vtss_packet_port_info_init, 578
 vtss_packet_ptp_action_t, 569
 vtss_packet_rx_conf_get, 572
 vtss_packet_rx_conf_set, 572
 vtss_packet_rx_frame_discard, 575
 vtss_packet_rx_frame_get, 574
 vtss_packet_rx_frame_get_raw, 575
 vtss_packet_rx_hdr_decode, 579
 vtss_packet_rx_hints_t, 569
 vtss_packet_rx_port_conf_get, 572
 vtss_packet_rx_port_conf_set, 574
 vtss_packet_tx_frame, 581
 vtss_packet_tx_frame_port, 576
 vtss_packet_tx_frame_port_vlan, 576
 vtss_packet_tx_frame_vlan, 577
 vtss_packet_tx_hdr_compile, 581
 vtss_packet_tx_hdr_encode, 579
 vtss_packet_tx_info_init, 582
 vtss_packet_tx_vstax_t, 570
 vtss_tag_type_t, 569
 vtss_packet_dma_conf_get
 vtss_packet_api.h, 582
 vtss_packet_dma_conf_set
 vtss_packet_api.h, 583
 vtss_packet_dma_conf_t, 142
 dma_enable, 142
 vtss_packet_dma_offset
 vtss_packet_api.h, 583
 vtss_packet_filter_t
 vtss_packet_api.h, 568
 vtss_packet_frame_filter
 vtss_packet_api.h, 577
 vtss_packet_frame_info_init
 vtss_packet_api.h, 577
 vtss_packet_frame_info_t, 143
 agr_rx_disable, 144
 agr_tx_disable, 144
 port_no, 143
 port_tx, 143
 vid, 143
 vtss_packet_oam_type_t
 vtss_packet_api.h, 568
 vtss_packet_port_filter_get
 vtss_packet_api.h, 578
 filter, 145
 tpid, 145
 vtss_packet_port_info_init
 vtss_packet_api.h, 578
 vtss_packet_port_info_t, 145
 agr_rx_disable, 146
 agr_tx_disable, 146
 port_no, 146
 vid, 146
 vtss_packet_ptp_action_t
 vtss_packet_api.h, 569
 vtss_packet_reg_type_t
 types.h, 399
 vtss_packet_rx_conf_get
 vtss_packet_api.h, 572
 vtss_packet_rx_conf_set
 vtss_packet_api.h, 572
 vtss_packet_rx_conf_t, 147
 grp_map, 148
 map, 147
 queue, 147
 reg, 147
 vtss_packet_rx_frame_discard
 vtss_packet_api.h, 575
 vtss_packet_rx_frame_get
 vtss_packet_api.h, 574
 vtss_packet_rx_frame_get_raw
 vtss_packet_api.h, 575
 vtss_packet_rx_grp_t
 types.h, 394
 vtss_packet_rx_hdr_decode
 vtss_packet_api.h, 579
 vtss_packet_rx_header_t, 148
 arrived_tagged, 149
 learn, 149
 length, 148
 port_no, 149
 queue_mask, 149
 tag, 149
 vtss_packet_rx_hints_t
 vtss_packet_api.h, 569
 vtss_packet_rx_info_t, 150
 acl_hit, 155
 acl_idx, 155
 cos, 154
 glag_no, 152
 hints, 151
 hw_tstamp, 156
 hw_tstamp_decoded, 157
 isdx, 158
 length, 151
 oam_info, 158
 oam_info_decoded, 158
 port_no, 151
 sflow_port_no, 157
 sflow_type, 157

stripped_tag, 153
sw_tstamp, 155
tag, 153
tag_type, 152
tstamp_id, 156
tstamp_id_decoded, 156
xtr_qu_mask, 154
vtss_packet_rx_meta_t, 159
chip_no, 160
etype, 161
fcs, 161
length, 162
no_wait, 160
sw_tstamp, 162
xtr_qu, 160
vtss_packet_rx_port_conf_get
 vtss_packet_api.h, 572
vtss_packet_rx_port_conf_set
 vtss_packet_api.h, 574
vtss_packet_rx_port_conf_t, 163
 bpdu_reg, 164
 garp_reg, 164
vtss_packet_rx_queue_conf_t, 164
 npi, 165
 size, 164
vtss_packet_rx_queue_map_t, 165
 bpdu_queue, 166
 garp_queue, 166
 igmp_queue, 166
 ipmc_ctrl_queue, 166
 learn_queue, 166
 lrn_all_queue, 167
 mac_vid_queue, 166
 sflow_queue, 167
 stack_queue, 167
vtss_packet_rx_queue_npi_conf_t, 167
 enable, 168
vtss_packet_rx_reg_t, 168
 bpdu_cpu_only, 169
 garp_cpu_only, 169
 igmp_cpu_only, 169
 ipmc_ctrl_cpu_copy, 169
 mld_cpu_only, 169
vtss_packet_tx_frame
 vtss_packet_api.h, 581
vtss_packet_tx_frame_port
 vtss_packet_api.h, 576
vtss_packet_tx_frame_port_vlan
 vtss_packet_api.h, 576
vtss_packet_tx_frame_vlan
 vtss_packet_api.h, 577
vtss_packet_tx_grp_t
 types.h, 394
vtss_packet_tx_hdr_compile
 vtss_packet_api.h, 581
vtss_packet_tx_hdr_encode
 vtss_packet_api.h, 579
vtss_packet_tx_ifh_t, 170
ifh, 170
length, 170
vtss_packet_tx_info_init
 vtss_packet_api.h, 582
vtss_packet_tx_info_t, 171
 agr_code, 174
 cos, 174
 dp, 178
 dst_port_mask, 172
 frm_len, 173
 isdx, 177
 isdx_dont_use, 178
 latch_timestamp, 176
 masquerade_port, 179
 oam_type, 177
 pdu_offset, 179
 ptp_action, 175
 ptp_id, 175
 ptp_timestamp, 176
 switch_frm, 172
 tag, 173
vtss_packet_tx_vstax_t
 vtss_packet_api.h, 570
vtss_phy_aneg_t, 180
 asymmetric_pause, 182
 speed_100m_fdx, 181
 speed_100m_hdx, 181
 speed_10m_fdx, 181
 speed_10m_hdx, 181
 speed_1g_fdx, 181
 speed_1g_hdx, 181
 symmetric_pause, 182
 tx_remote_fault, 182
vtss_phy_api.h
 lb_type, 614
 MAX_CFG_BUF_SIZE, 594
 MAX_STAT_BUF_SIZE, 594
 MAX_WOL_MAC_ADDR_SIZE, 605
 MAX_WOL_PASSWD_SIZE, 605
 MIN_WOL_PASSWD_SIZE, 605
 rgmii_skew_delay_psec_t, 608
 VTSS_PHY_ACTIPHY_PWR, 595
 VTSS_PHY_LINK_AMS_EV, 600
 VTSS_PHY_LINK_AUTO_NEG_COMPLETE_EV,
 600
 VTSS_PHY_LINK_AUTO_NEG_ERROR_EV, 600
 VTSS_PHY_LINK_DOWN_PWR, 595
 VTSS_PHY_LINK_EXT_EEE_LINKFAIL_EV, 603
 VTSS_PHY_LINK_EXT_EEE_WAIT_RX_TQ_EV,
 603
 VTSS_PHY_LINK_EXT_EEE_WAIT_TS_EV, 603
 VTSS_PHY_LINK_EXT_EEE_WAKE_ERR_EV,
 603
 VTSS_PHY_LINK_EXT_MACSEC_EGRESS_EV,
 604
 VTSS_PHY_LINK_EXT_MACSEC_FC_BUFF←
 EV, 604

VTSS_PHY_LINK_EXT_MACSEC_HOST_MAC←
 _EV, 604
 VTSS_PHY_LINK_EXT_MACSEC_INGRESS←
 _EV, 604
 VTSS_PHY_LINK_EXT_MACSEC_LINE_MAC←
 _EV, 604
 VTSS_PHY_LINK_EXT_MEM_INT_RING_EV,
 605
 VTSS_PHY_LINK_EXT_RR_SW_COMPL_EV,
 603
 VTSS_PHY_LINK_EXTENDED_REG_INT_EV,
 602
 VTSS_PHY_LINK_FALSE_CARRIER_INT_EV,
 601
 VTSS_PHY_LINK_FDX_STATE_CHANGE_EV,
 600
 VTSS_PHY_LINK_FFAIL_EV, 599
 VTSS_PHY_LINK_INLINE_POW_DEV_DETECT←
 _T_EV, 601
 VTSS_PHY_LINK_LINK_SPEED_DS_DETECT←
 _EV, 602
 VTSS_PHY_LINK_LOS_EV, 599
 VTSS_PHY_LINK_MASTER_SLAVE_RES_ER←
 _R_EV, 602
 VTSS_PHY_LINK_RX_ER_INT_EV, 602
 VTSS_PHY_LINK_RX_FIFO_OVERFLOW_INT←
 _EV, 601
 VTSS_PHY_LINK_SPEED_STATE_CHANGE←
 _EV, 600
 VTSS_PHY_LINK_SYMBOL_ERR_INT_EV, 601
 VTSS_PHY_LINK_TX_FIFO_OVERFLOW_INT←
 _EV, 601
 VTSS_PHY_LINK_UP_FULL_PWR, 595
 VTSS_PHY_LINK_WAKE_ON_LAN_INT_EV, 602
 VTSS_PHY_PAGE_0x2DAF, 598
 VTSS_PHY_PAGE_1588, 597
 VTSS_PHY_PAGE_EXTENDED_2, 596
 VTSS_PHY_PAGE_EXTENDED_3, 597
 VTSS_PHY_PAGE_EXTENDED_4, 597
 VTSS_PHY_PAGE_EXTENDED, 596
 VTSS_PHY_PAGE_GPIO, 597
 VTSS_PHY_PAGE_MACSEC, 597
 VTSS_PHY_PAGE_STANDARD, 596
 VTSS_PHY_PAGE_TEST, 598
 VTSS_PHY_PAGE_TR, 598
 VTSS_PHY_POWER_ACTIPHY_BIT, 594
 VTSS_PHY_POWER_DYNAMIC_BIT, 595
 VTSS_PHY_RECov_CLK1, 595
 VTSS_PHY_RECov_CLK2, 596
 VTSS_PHY_RECov_CLK_NUM, 596
 VTSS_PHY_REG_EXTENDED, 598
 VTSS_PHY_REG_GPIO, 599
 VTSS_PHY_REG_STANDARD, 598
 VTSS_PHY_REG_TEST, 599
 VTSS_PHY_REG_TR, 599
 vga_adc_debug, 634
 vtss_fefi_mode_t, 615
 vtss_phy_atom12_patch_settings_get, 644
 vtss_phy_cfg_ib_cterm, 644
 vtss_phy_cfg_ob_post0, 643
 vtss_phy_chip_temp_get, 617
 vtss_phy_chip_temp_init, 618
 vtss_phy_cl37_lp_abil_get, 619
 vtss_phy_clk_source_t, 613
 vtss_phy_clk_squelch, 613
 vtss_phy_clock_conf_get, 623
 vtss_phy_clock_conf_set, 623
 vtss_phy_coma_mode_disable, 633
 vtss_phy_coma_mode_enable, 634
 vtss_phy_conf_1g_get, 620
 vtss_phy_conf_1g_set, 621
 vtss_phy_conf_get, 618
 vtss_phy_conf_set, 619
 vtss_phy_csr_rd, 639
 vtss_phy_csr_wr, 639
 vtss_phy_debug_phyinfo_print, 647
 vtss_phy_debug_regdump_print, 649
 vtss_phy_debug_register_dump, 647
 vtss_phy_debug_stat_print, 646
 vtss_phy_detect_base_ports, 648
 vtss_phy_do_page_chk_get, 642
 vtss_phy_do_page_chk_set, 641
 vtss_phy_eee_conf_get, 635
 vtss_phy_eee_conf_set, 636
 vtss_phy_eee_ena, 635
 vtss_phy_eee_link_partner_advertisements_get,
 637
 vtss_phy_eee_power_save_state_get, 636
 vtss_phy_enhanced_led_control_init, 632
 vtss_phy_enhanced_led_control_init_get, 633
 vtss_phy_event_enable_get, 637
 vtss_phy_event_enable_set, 637
 vtss_phy_event_poll, 638
 vtss_phy_ext_connector_loopback, 648
 vtss_phy_ext_event_poll, 659
 vtss_phy_fast_link_fail_t, 609
 vtss_phy_fefi_detect, 657
 vtss_phy_fefi_get, 656
 vtss_phy_fefi_set, 656
 vtss_phy_flowcontrol_decode_status, 645
 vtss_phy_forced_reset_t, 608
 vtss_phy_freq_t, 613
 vtss_phy_gpio_get, 629
 vtss_phy_gpio_mode, 629
 vtss_phy_gpio_mode_t, 614
 vtss_phy_gpio_set, 630
 vtss_phy_i2c_read, 624
 vtss_phy_i2c_write, 625
 vtss_phy_id_get, 620
 vtss_phy_is_8051_crc_ok, 643
 vtss_phy_is_viper_revB, 659
 vtss_phy_lcpll_status_get, 653
 vtss_phy_led_intensity, 606
 vtss_phy_led_intensity_get, 632
 vtss_phy_led_intensity_set, 631
 vtss_phy_led_mode_set, 631

vtss_phy_led_mode_t, 606
vtss_phy_loopback_get, 642
vtss_phy_loopback_set, 642
vtss_phy_mac_media_inhibit_odd_start, 655
vtss_phy_mac_serdes_pcs_sgmii_pre, 610
vtss_phy_macsec_csr_sd6g_rd, 660
vtss_phy_macsec_csr_sd6g_wr, 660
vtss_phy_mdi_t, 608
vtss_phy_media_force_ams_sel_t, 612
vtss_phy_media_interface_t, 607
vtss_phy_media_rem_fault_t, 612
vtss_phy_mmd_read, 626
vtss_phy_mmd_write, 627
vtss_phy_mode_t, 609
vtss_phy_mse_1000m_get, 658
vtss_phy_mse_100m_get, 657
vtss_phy_patch_settings_get, 651
vtss_phy_pkt_mode_t, 609
vtss_phy_port_eee_capable, 634
vtss_phy_post_reset, 616
vtss_phy_power_conf_get, 622
vtss_phy_power_conf_set, 622
vtss_phy_power_status_get, 622
vtss_phy_pre_reset, 615
vtss_phy_pre_system_reset, 616
vtss_phy_read, 625
vtss_phy_read_page, 626
vtss_phy_read_tr_addr, 658
vtss_phy_recov_clk_t, 606
vtss_phy_reg_decode_status, 645
vtss_phy_reset, 616
vtss_phy_reset_get, 617
vtss_phy_reset_lcpll, 653
vtss_phy_sd6g_mac_serdes_conf, 661
vtss_phy_sd6g_ob_lev_rd, 655
vtss_phy_sd6g_ob_lev_wr, 655
vtss_phy_sd6g_ob_post_rd, 654
vtss_phy_sd6g_ob_post_wr, 654
vtss_phy_serdes1g_rcpll_status_get, 652
vtss_phy_serdes6g_rcpll_status_get, 652
vtss_phy_serdes_fmedia_loopback, 649
vtss_phy_serdes_sgmii_loopback, 648
vtss_phy_sigdet_polarity_t, 610
vtss_phy_statistic_get, 641
vtss_phy_status_1g_get, 621
vtss_phy_status_get, 619
vtss_phy_status_inst_poll, 660
vtss_phy_unidirectional_t, 610
vtss_phy_veriphy_get, 631
vtss_phy_veriphy_start, 630
vtss_phy_warm_start_failed_get, 646
vtss_phy_wol_conf_get, 650
vtss_phy_wol_conf_set, 651
vtss_phy_wol_enable, 650
vtss_phy_write, 627
vtss_phy_write_masked, 628
vtss_phy_write_masked_page, 628
vtss_squelch_workaround, 638
vtss_wol_passwd_len_type_t, 615
vtss_phy_atom12_patch_settings_get
 vtss_phy_api.h, 644
vtss_phy_cfg_ib_cterm
 vtss_phy_api.h, 644
vtss_phy_cfg_ob_post0
 vtss_phy_api.h, 643
vtss_phy_chip_temp_get
 vtss_phy_api.h, 617
vtss_phy_chip_temp_init
 vtss_phy_api.h, 618
vtss_phy_cl37_lp_abil_get
 vtss_phy_api.h, 619
vtss_phy_clk_source_t
 vtss_phy_api.h, 613
vtss_phy_clk_squelch
 vtss_phy_api.h, 613
vtss_phy_clock_conf_get
 vtss_phy_api.h, 623
vtss_phy_clock_conf_set
 vtss_phy_api.h, 623
vtss_phy_clock_conf_t, 182
 freq, 183
 squelch, 183
 src, 183
vtss_phy_coma_mode_disable
 vtss_phy_api.h, 633
vtss_phy_coma_mode_enable
 vtss_phy_api.h, 634
vtss_phy_conf_1g_get
 vtss_phy_api.h, 620
vtss_phy_conf_1g_set
 vtss_phy_api.h, 621
vtss_phy_conf_1g_t, 184
 cfg, 184
 master, 184
 val, 184
vtss_phy_conf_get
 vtss_phy_api.h, 618
vtss_phy_conf_set
 vtss_phy_api.h, 619
vtss_phy_conf_t, 185
 aneg, 186
 flf, 186
 force_ams_sel, 187
 forced, 185
 mac_if_pcs, 187
 mdi, 186
 media_if_pcs, 187
 mode, 185
 sigdet, 186
 skip_coma, 187
 unidir, 186
vtss_phy_csr_rd
 vtss_phy_api.h, 639
vtss_phy_csr_wr
 vtss_phy_api.h, 639
vtss_phy_debug_phyinfo_print

vtss_phy_api.h, 647
 vtss_phy_debug_regdump_print
 vtss_phy_api.h, 649
 vtss_phy_debug_register_dump
 vtss_phy_api.h, 647
 vtss_phy_debug_stat_print
 vtss_phy_api.h, 646
 vtss_phy_detect_base_ports
 vtss_phy_api.h, 648
 vtss_phy_do_page_chk_get
 vtss_phy_api.h, 642
 vtss_phy_do_page_chk_set
 vtss_phy_api.h, 641
 vtss_phy_eee_conf_get
 vtss_phy_api.h, 635
 vtss_phy_eee_conf_set
 vtss_phy_api.h, 636
 vtss_phy_eee_conf_t, 188
 eee_ena_phy, 188
 eee_mode, 188
 vtss_phy_eee_ena
 vtss_phy_api.h, 635
 vtss_phy_eee_link_partner_advertisements_get
 vtss_phy_api.h, 637
 vtss_phy_eee_power_save_state_get
 vtss_phy_api.h, 636
 vtss_phy_enhanced_led_control_init
 vtss_phy_api.h, 632
 vtss_phy_enhanced_led_control_init_get
 vtss_phy_api.h, 633
 vtss_phy_enhanced_led_control_t, 188
 ser_led_frame_rate, 189
 ser_led_output_1, 189
 ser_led_output_2, 189
 ser_led_select, 189
 vtss_phy_event_enable_get
 vtss_phy_api.h, 637
 vtss_phy_event_enable_set
 vtss_phy_api.h, 637
 vtss_phy_event_poll
 vtss_phy_api.h, 638
 vtss_phy_ext_connector_loopback
 vtss_phy_api.h, 648
 vtss_phy_ext_event_poll
 vtss_phy_api.h, 659
 vtss_phy_fast_link_fail_t
 vtss_phy_api.h, 609
 vtss_phy_fefi_detect
 vtss_phy_api.h, 657
 vtss_phy_fefi_get
 vtss_phy_api.h, 656
 vtss_phy_fefi_set
 vtss_phy_api.h, 656
 vtss_phy_flowcontrol_decode_status
 vtss_phy_api.h, 645
 vtss_phy_forced_reset_t
 vtss_phy_api.h, 608
 vtss_phy_forced_t, 190
 fdx, 190
 speed, 190
 vtss_phy_freq_t
 vtss_phy_api.h, 613
 vtss_phy_gpio_get
 vtss_phy_api.h, 629
 vtss_phy_gpio_mode
 vtss_phy_api.h, 629
 vtss_phy_gpio_mode_t
 vtss_phy_api.h, 614
 vtss_phy_gpio_set
 vtss_phy_api.h, 630
 vtss_phy_i2c_read
 vtss_phy_api.h, 624
 vtss_phy_i2c_write
 vtss_phy_api.h, 625
 vtss_phy_id_get
 vtss_phy_api.h, 620
 vtss_phy_is_8051_crc_ok
 vtss_phy_api.h, 643
 vtss_phy_is_viper_revB
 vtss_phy_api.h, 659
 vtss_phy_lcpll_status_get
 vtss_phy_api.h, 653
 vtss_phy_led_intensity
 vtss_phy_api.h, 606
 vtss_phy_led_intensity_get
 vtss_phy_api.h, 632
 vtss_phy_led_intensity_set
 vtss_phy_api.h, 631
 vtss_phy_led_mode_select_t, 191
 mode, 191
 number, 191
 vtss_phy_led_mode_set
 vtss_phy_api.h, 631
 vtss_phy_led_mode_t
 vtss_phy_api.h, 606
 vtss_phy_loopback_get
 vtss_phy_api.h, 642
 vtss_phy_loopback_set
 vtss_phy_api.h, 642
 vtss_phy_loopback_t, 192
 connector_enable, 193
 far_end_enable, 192
 mac_serdes_equipment_enable, 193
 mac_serdes_facility_enable, 193
 mac_serdes_input_enable, 193
 media_serdes_equipment_enable, 194
 media_serdes_facility_enable, 194
 media_serdes_input_enable, 193
 near_end_enable, 192
 vtss_phy_mac_media_inhibit_odd_start
 vtss_phy_api.h, 655
 vtss_phy_mac_serd_pcs_cntl_t, 194
 aneg_restart, 195
 disable, 195
 fast_link_stat_ena, 197
 force_adv_ability, 195

inhibit_odd_start, 197
pd_enable, 195
restart, 195
serdes_aneg_ena, 196
serdes_pol_inv_in, 196
serdes_pol_inv_out, 196
sgmii_in_pre, 196
sgmii_out_pre, 196
vtss_phy_mac_serdes_pcs_sgmii_pre
 vtss_phy_api.h, 610
vtss_phy_macsec_csr_sd6g_rd
 vtss_phy_api.h, 660
vtss_phy_macsec_csr_sd6g_wr
 vtss_phy_api.h, 660
vtss_phy_mdi_t
 vtss_phy_api.h, 608
vtss_phy_media_force_ams_sel_t
 vtss_phy_api.h, 612
vtss_phy_media_interface_t
 vtss_phy_api.h, 607
vtss_phy_media_rem_fault_t
 vtss_phy_api.h, 612
vtss_phy_media_serdes_pcs_ctrl_t, 197
 aneg_pd_detect, 198
 force_adv_ability, 198
 force_fefi, 199
 force_fefi_value, 199
 force_hls, 199
 inhibit_odd_start, 199
 remote_fault, 198
 serdes_pol_inv_in, 198
 serdes_pol_inv_out, 198
vtss_phy_mmd_read
 vtss_phy_api.h, 626
vtss_phy_mmd_write
 vtss_phy_api.h, 627
vtss_phy_mode_t
 vtss_phy_api.h, 609
vtss_phy_mse_1000m_get
 vtss_phy_api.h, 658
vtss_phy_mse_100m_get
 vtss_phy_api.h, 657
vtss_phy_patch_settings_get
 vtss_phy_api.h, 651
vtss_phy_pkt_mode_t
 vtss_phy_api.h, 609
vtss_phy_port_eee_capable
 vtss_phy_api.h, 634
vtss_phy_post_reset
 vtss_phy_api.h, 616
vtss_phy_power_conf_get
 vtss_phy_api.h, 622
vtss_phy_power_conf_set
 vtss_phy_api.h, 622
vtss_phy_power_conf_t, 200
 mode, 200
vtss_phy_power_mode_t
 phy.h, 352
vtss_phy_power_status_get
 vtss_phy_api.h, 622
vtss_phy_power_status_t, 200
 level, 201
vtss_phy_pre_reset
 vtss_phy_api.h, 615
vtss_phy_pre_system_reset
 vtss_phy_api.h, 616
vtss_phy_read
 vtss_phy_api.h, 625
vtss_phy_read_page
 vtss_phy_api.h, 626
vtss_phy_read_tr_addr
 vtss_phy_api.h, 658
vtss_phy_recov_clk_t
 vtss_phy_api.h, 606
vtss_phy_reg_decode_status
 vtss_phy_api.h, 645
vtss_phy_reset
 vtss_phy_api.h, 616
vtss_phy_reset_conf_t, 201
 force, 202
 i_cpu_en, 203
 mac_if, 202
 media_if, 202
 pkt_mode, 202
 rgmii, 202
 tbi, 202
vtss_phy_reset_get
 vtss_phy_api.h, 617
vtss_phy_reset_lcpll
 vtss_phy_api.h, 653
vtss_phy_rgmii_conf_t, 203
 rx_clk_skew_ps, 203
 tx_clk_skew_ps, 204
vtss_phy_sd6g_mac_serdes_conf
 vtss_phy_api.h, 661
vtss_phy_sd6g_ob_lev_rd
 vtss_phy_api.h, 655
vtss_phy_sd6g_ob_lev_wr
 vtss_phy_api.h, 655
vtss_phy_sd6g_ob_post_rd
 vtss_phy_api.h, 654
vtss_phy_sd6g_ob_post_wr
 vtss_phy_api.h, 654
vtss_phy_serdes1g_rcpll_status_get
 vtss_phy_api.h, 652
vtss_phy_serdes6g_rcpll_status_get
 vtss_phy_api.h, 652
vtss_phy_serdes_fmedia_loopback
 vtss_phy_api.h, 649
vtss_phy_serdes_sgmii_loopback
 vtss_phy_api.h, 648
vtss_phy_sigdet_polarity_t
 vtss_phy_api.h, 610
vtss_phy_statistic_get
 vtss_phy_api.h, 641
vtss_phy_statistic_t, 204

cu_bad, 205
 cu_good, 204
 media_mac_serdes_crc, 206
 media_mac_serdes_good, 205
 rx_err_cnt_base_tx, 205
 serdes_tx_bad, 205
 serdes_tx_good, 205
 vtss_phy_status_1g_get
 vtss_phy_api.h, 621
 vtss_phy_status_1g_t, 206
 master, 207
 master_cfg_fault, 206
 vtss_phy_status_get
 vtss_phy_api.h, 619
 vtss_phy_status_inst_poll
 vtss_phy_api.h, 660
 vtss_phy_tbi_conf_t, 207
 aneg_enable, 207
 vtss_phy_type_t, 208
 base_port_no, 209
 channel_id, 209
 part_number, 208
 phy_api_base_no, 209
 port_cnt, 209
 revision, 208
 vtss_phy_unidirectional_t
 vtss_phy_api.h, 610
 vtss_phy_veriphy_get
 vtss_phy_api.h, 631
 vtss_phy_veriphy_result_t, 210
 length, 210
 link, 210
 status, 210
 vtss_phy_veriphy_start
 vtss_phy_api.h, 630
 vtss_phy_veriphy_status_t
 phy.h, 353
 vtss_phy_warm_start_failed_get
 vtss_phy_api.h, 646
 vtss_phy_wol_conf_get
 vtss_phy_api.h, 650
 vtss_phy_wol_conf_set
 vtss_phy_api.h, 651
 vtss_phy_wol_conf_t, 211
 magic_pkt_cnt, 212
 secure_on_enable, 211
 wol_mac, 211
 wol_pass, 212
 wol_passwd_len, 212
 vtss_phy_wol_enable
 vtss_phy_api.h, 650
 vtss_phy_write
 vtss_phy_api.h, 627
 vtss_phy_write_masked
 vtss_phy_api.h, 628
 vtss_phy_write_masked_page
 vtss_phy_api.h, 628
 vtss_pi_conf_t, 212
 cs_wait_ns, 213
 use_extended_bus_cycle, 213
 width, 213
 vtss_policer_ext_t, 214
 flow_control, 214
 frame_rate, 214
 vtss_policer_t, 214
 level, 215
 rate, 215
 vtss_policer_type_t
 types.h, 399
 vtss_poll_1sec
 vtss_misc_api.h, 523
 vtss_port_api.h
 CHIP_PORT_UNUSED, 664
 VTSS_FRAME_GAP_DEFAULT, 664
 VTSS_MAX_FRAME_LENGTH_MAX, 665
 VTSS_MAX_FRAME_LENGTH_STANDARD, 664
 vtss_internal_bw_t, 665
 vtss_miim_controller_t, 665
 vtss_miim_read, 673
 vtss_miim_write, 673
 vtss_port_basic_counters_get, 672
 vtss_port_clause_37_control_get, 668
 vtss_port_clause_37_control_set, 669
 vtss_port_clause_37_remote_fault_t, 666
 vtss_port_conf_get, 669
 vtss_port_conf_set, 669
 vtss_port_counters_clear, 671
 vtss_port_counters_get, 671
 vtss_port_counters_update, 670
 vtss_port_forward_state_get, 672
 vtss_port_forward_state_set, 672
 vtss_port_forward_t, 667
 vtss_port_loop_t, 667
 vtss_port_map_get, 668
 vtss_port_map_set, 667
 vtss_port_max_tags_t, 666
 vtss_port_status_get, 670
 vtss_port_basic_counters_get
 vtss_port_api.h, 672
 vtss_port_bridge_counters_t, 215
 dot1dTpPortInDiscards, 216
 vtss_port_clause_37_adv_t, 216
 acknowledge, 217
 asymmetric_pause, 217
 fdx, 217
 hdx, 217
 next_page, 218
 remote_fault, 217
 symmetric_pause, 217
 vtss_port_clause_37_control_get
 vtss_port_api.h, 668
 vtss_port_clause_37_control_set
 vtss_port_api.h, 669
 vtss_port_clause_37_control_t, 218
 advertisement, 219
 enable, 218

vtss_port_clause_37_remote_fault_t
 vtss_port_api.h, 666

vtss_port_conf_get
 vtss_port_api.h, 669

vtss_port_conf_set
 vtss_port_api.h, 669

vtss_port_conf_t, 219

 exc_col_cont, 222

 fdx, 221

 flow_control, 221

 frame_gaps, 220

 frame_length_chk, 222

 if_type, 220

 loop, 223

 max_frame_length, 221

 max_tags, 222

 power_down, 221

 sd_active_high, 220

 sd_enable, 220

 sd_internal, 220

 serdes, 223

 speed, 221

 xaui_rx_lane_flip, 222

 xaui_tx_lane_flip, 222

vtss_port_counters_clear
 vtss_port_api.h, 671

vtss_port_counters_get
 vtss_port_api.h, 671

vtss_port_counters_t, 223

 bridge, 224

 ethernet_like, 224

 evc, 225

 if_group, 224

 prop, 224

 rmon, 224

vtss_port_counters_update
 vtss_port_api.h, 670

vtss_port_ethernet_like_counters_t, 225

 dot3InPauseFrames, 225

 dot3OutPauseFrames, 226

vtss_port_evc_counters_t, 226

 rx_green, 226

 rx_green_discard, 227

 rx_red, 227

 rx_yellow, 227

 rx_yellow_discard, 227

 tx_green, 227

 tx_yellow, 228

vtss_port_flow_control_conf_t, 228

 generate, 229

 obey, 228

 pfc, 229

 smac, 229

vtss_port_forward_state_get
 vtss_port_api.h, 672

vtss_port_forward_state_set
 vtss_port_api.h, 672

vtss_port_forward_t

 vtss_port_api.h, 667

vtss_port_frame_gaps_t, 229

 fdx_gap, 230

 hdx_gap_1, 230

 hdx_gap_2, 230

vtss_port_if_group_counters_t, 231

 ifInBroadcastPkts, 232

 ifInDiscards, 232

 ifInErrors, 232

 ifInMulticastPkts, 231

 ifInNUcastPkts, 232

 ifInOctets, 231

 ifInUcastPkts, 231

 ifOutBroadcastPkts, 233

 ifOutDiscards, 233

 ifOutErrors, 234

 ifOutMulticastPkts, 233

 ifOutNUcastPkts, 233

 ifOutOctets, 232

 ifOutUcastPkts, 233

vtss_port_interface_t
 types.h, 397

vtss_port_loop_t
 vtss_port_api.h, 667

vtss_port_map_get
 vtss_port_api.h, 668

vtss_port_map_set
 vtss_port_api.h, 667

vtss_port_map_t, 234

 chip_no, 235

 chip_port, 234

 miim_addr, 235

 miim_chip_no, 235

 miim_controller, 235

vtss_port_max_tags_t
 vtss_port_api.h, 666

vtss_port_mux_mode_t
 vtss_init_api.h, 442

vtss_port_proprietary_counters_t, 236

 rx_prio, 236

 tx_prio, 236

vtss_port_rmon_counters_t, 236

 rx_etherStatsBroadcastPkts, 238

 rx_etherStatsCRCAlignErrors, 238

 rx_etherStatsDropEvents, 237

 rx_etherStatsFragments, 239

 rx_etherStatsJabbers, 239

 rx_etherStatsMulticastPkts, 238

 rx_etherStatsOctets, 237

 rx_etherStatsOversizePkts, 239

 rx_etherStatsPkts, 238

 rx_etherStatsPkts1024to1518Octets, 240

 rx_etherStatsPkts128to255Octets, 240

 rx_etherStatsPkts1519toMaxOctets, 240

 rx_etherStatsPkts256to511Octets, 240

 rx_etherStatsPkts512to1023Octets, 240

 rx_etherStatsPkts64Octets, 239

 rx_etherStatsPkts65to127Octets, 239

rx_etherStatsUndersizePkts, 238
 tx_etherStatsBroadcastPkts, 241
 tx_etherStatsCollisions, 242
 tx_etherStatsDropEvents, 241
 tx_etherStatsMulticastPkts, 241
 tx_etherStatsOctets, 241
 tx_etherStatsPkts, 241
 tx_etherStatsPkts1024to1518Octets, 243
 tx_etherStatsPkts128to255Octets, 242
 tx_etherStatsPkts1519toMaxOctets, 243
 tx_etherStatsPkts256to511Octets, 242
 tx_etherStatsPkts512to1023Octets, 243
 tx_etherStatsPkts64Octets, 242
 tx_etherStatsPkts65to127Octets, 242
 vtss_port_serdes_conf_t, 243
 sfp_dac, 244
 vtss_port_sgmii_aneg_t, 244
 aneg_complete, 246
 fdx, 245
 hdx, 245
 link, 245
 speed_100M, 245
 speed_10M, 245
 speed_1G, 245
 vtss_port_speed_t
 port.h, 365
 vtss_port_state_get
 vtss_l2_api.h, 469
 vtss_port_state_set
 vtss_l2_api.h, 469
 vtss_port_status_get
 vtss_port_api.h, 670
 vtss_port_status_t, 246
 aneg, 248
 aneg_complete, 247
 copper, 248
 fdx, 247
 fiber, 248
 link, 247
 link_down, 247
 mdi_cross, 248
 remote_fault, 247
 speed, 247
 unidirectional_ability, 248
 vtss_ptp_event_enable
 vtss_misc_api.h, 524
 vtss_ptp_event_poll
 vtss_misc_api.h, 523
 vtss_pvlan_port_members_get
 vtss_l2_api.h, 482
 vtss_pvlan_port_members_set
 vtss_l2_api.h, 483
 vtss_qce_action_t, 249
 dei, 251
 dp, 250
 dp_enable, 250
 dscp, 250
 dscp_enable, 250
 pcp, 251
 pcp_dei_enable, 250
 policy_no, 251
 policy_no_enable, 251
 prio, 249
 prio_enable, 249
 vtss_qce_add
 vtss_qos_api.h, 682
 vtss_qce_del
 vtss_qos_api.h, 683
 vtss_qce_frame_etype_t, 252
 data, 252
 etype, 252
 vtss_qce_frame_ipv4_t, 252
 dport, 254
 dscp, 253
 fragment, 253
 proto, 253
 sip, 253
 sport, 254
 vtss_qce_frame_ipv6_t, 254
 dport, 255
 dscp, 255
 proto, 255
 sip, 255
 sport, 255
 vtss_qce_frame_llc_t, 256
 data, 256
 vtss_qce_frame_snap_t, 257
 data, 257
 vtss_qce_init
 vtss_qos_api.h, 682
 vtss_qce_key_t, 257
 etype, 258
 frame, 259
 ipv4, 259
 ipv6, 259
 llc, 259
 mac, 258
 port_list, 258
 snap, 259
 tag, 258
 type, 258
 vtss_qce_mac_t, 260
 dmac_bc, 260
 dmac_mc, 260
 smac, 261
 vtss_qce_t, 261
 action, 262
 id, 261
 key, 262
 vtss_qce_tag_t, 262
 dei, 263
 pcp, 263
 s_tag, 263
 tagged, 263
 vid, 263
 vtss_qce_type_t

vtss_qos_api.h, 679
vtss_qos_api.h
 VTSS_PORT_POLICER_CPU_QUEUES, 676
 VTSS_PORT_POLICERS, 676
 VTSS_QCE_ID_LAST, 677
 VTSS_QCL_ARRAY_SIZE, 677
 VTSS_QCL_ID_END, 677
 VTSS_QCL_ID_START, 677
 VTSS_QCL_IDS, 677
 vtss_dscp_emode_t, 678
 vtss_dscp_mode_t, 678
 vtss_mep_policer_conf_get, 680
 vtss_mep_policer_conf_set, 680
 vtss_qce_add, 682
 vtss_qce_del, 683
 vtss_qce_init, 682
 vtss_qce_type_t, 679
 vtss_qos_conf_get, 679
 vtss_qos_conf_set, 680
 vtss_qos_port_conf_get, 681
 vtss_qos_port_conf_set, 681
 vtss_tag_remark_mode_t, 678
vtss_qos_conf_get
 vtss_qos_api.h, 679
vtss_qos_conf_set
 vtss_qos_api.h, 680
vtss_qos_conf_t, 264
 dscp_dp_level_map, 265
 dscp_qos_class_map, 265
 dscp_qos_map, 265
 dscp_qos_map_dp1, 265
 dscp_remap, 266
 dscp_remap_dp1, 266
 dscp_remark, 266
 dscp_translate_map, 266
 dscp_trust, 265
 policer_bc, 268
 policer_bc_frame_rate, 268
 policer_bc_mode, 268
 policer_mc, 267
 policer_mc_frame_rate, 267
 policer_mc_mode, 267
 policer_uc, 266
 policer_uc_frame_rate, 267
 policer_uc_mode, 267
 prios, 264
vtss_qos_port_conf_get
 vtss_qos_api.h, 681
vtss_qos_port_conf_set
 vtss_qos_api.h, 681
vtss_qos_port_conf_t, 268
 default_dei, 271
 default_dpl, 271
 default_prio, 270
 dmac_dip, 274
 dp_level_map, 272
 dscp_class_enable, 272
 dscp_emode, 272
 dscp_mode, 272
 dscp_translate, 272
 dwrr_enable, 274
 excess_enable, 270
 policer_ext_port, 269
 policer_port, 269
 policer_queue, 270
 qos_class_map, 271
 queue_pct, 274
 shaper_port, 270
 shaper_queue, 270
 tag_class_enable, 271
 tag_default_dei, 273
 tag_default_pcp, 273
 tag_dei_map, 273
 tag_pcp_map, 273
 tag_remark_mode, 273
 usr_prio, 271
vtss_rcpl_status_t, 274
 cal_error, 275
 cal_not_done, 275
 out_of_range, 275
vtss_reg_read
 vtss_misc_api.h, 521
vtss_reg_read_t
 vtss_init_api.h, 435
vtss_reg_write
 vtss_misc_api.h, 521
vtss_reg_write_masked
 vtss_misc_api.h, 522
vtss_reg_write_t
 vtss_init_api.h, 436
vtss_restart_conf_end
 vtss_init_api.h, 445
vtss_restart_conf_get
 vtss_init_api.h, 445
vtss_restart_conf_set
 vtss_init_api.h, 446
vtss_restart_status_get
 vtss_init_api.h, 445
vtss_restart_status_t, 276
 cur_version, 276
 prev_version, 276
 restart, 276
vtss_restart_t
 vtss_init_api.h, 442
vtss_routing_entry_t, 277
 ipv4_uc, 277
 ipv6_uc, 278
 route, 278
 type, 277
 vlan, 278
vtss_rx_master_timestamp_get
 vtss_ts_api.h, 712
vtss_rx_timestamp_get
 vtss_ts_api.h, 711
vtss_rx_timestamp_id_release
 vtss_ts_api.h, 712

vtss_secure_on_passwd_t, 278
 passwd, 279
 vtss_security_api.h
 VTSS_ACE_ID_LAST, 686
 vtss_ace_add, 692
 vtss_ace_bit_t, 688
 vtss_ace_counter_clear, 693
 vtss_ace_counter_get, 693
 vtss_ace_del, 692
 vtss_ace_init, 691
 vtss_ace_type_t, 687
 vtss_acl_policer_conf_get, 689
 vtss_acl_policer_conf_set, 689
 vtss_acl_port_action_t, 687
 vtss_acl_port_conf_get, 690
 vtss_acl_port_conf_set, 690
 vtss_acl_port_counter_clear, 691
 vtss_acl_port_counter_get, 691
 vtss_acl_ptp_action_t, 687
 vtss_auth_port_state_get, 688
 vtss_auth_port_state_set, 688
 vtss_auth_state_t, 686
 vtss_serdes_macro_conf_t, 279
 ib_cterm_ena, 280
 qsgmii, 280
 serdes1g_vdd, 279
 serdes6g_vdd, 280
 vtss_serdes_mode_t
 types.h, 398
 vtss_sflow_port_conf_get
 vtss_l2_api.h, 485
 vtss_sflow_port_conf_set
 vtss_l2_api.h, 485
 vtss_sflow_port_conf_t, 280
 sampling_rate, 281
 type, 281
 vtss_sflow_sampling_rate_convert
 vtss_l2_api.h, 486
 vtss_sflow_type_t
 l2_types.h, 333
 vtss_sgpi_bmode_t
 vtss_misc_api.h, 516
 vtss_sgpi_conf_get
 vtss_misc_api.h, 528
 vtss_sgpi_conf_set
 vtss_misc_api.h, 528
 vtss_sgpi_conf_t, 281
 bit_count, 282
 bmode, 282
 port_conf, 282
 vtss_sgpi_event_enable
 vtss_misc_api.h, 530
 vtss_sgpi_event_poll
 vtss_misc_api.h, 529
 vtss_sgpi_mode_t
 vtss_misc_api.h, 515
 vtss_sgpi_port_conf_t, 283
 enabled, 283
 int_pol_high, 283
 mode, 283
 vtss_sgpi_port_data_t, 284
 value, 284
 vtss_sgpi_read
 vtss_misc_api.h, 529
 vtss_shaper_t, 285
 level, 285
 rate, 285
 vtss_spi_32bit_read_write_t
 vtss_init_api.h, 438
 vtss_spi_64bit_read_write_t
 vtss_init_api.h, 438
 vtss_spi_read_write_t
 vtss_init_api.h, 437
 vtss_squelch_workaround
 vtss_phy_api.h, 638
 vtss_storm_policer_mode_t
 types.h, 399
 vtss_stp_port_state_get
 vtss_l2_api.h, 469
 vtss_stp_port_state_set
 vtss_l2_api.h, 470
 vtss_stp_state_t
 vtss_l2_api.h, 460
 vtss_sync_api.h
 VTSS_SYNCE_CLK_MAX, 695
 VTSS_SYNCE_CLK_A, 695
 VTSS_SYNCE_CLK_B, 695
 vtss_sync_clock_in_get, 697
 vtss_sync_clock_in_set, 697
 vtss_sync_clock_out_get, 696
 vtss_sync_clock_out_set, 696
 vtss_sync_divider_t, 695
 vtss_sync_clock_in_get
 vtss_sync_api.h, 697
 vtss_sync_clock_in_set
 vtss_sync_api.h, 697
 vtss_sync_clock_in_t, 286
 enable, 286
 port_no, 286
 squelsh, 286
 vtss_sync_clock_out_get
 vtss_sync_api.h, 696
 vtss_sync_clock_out_set
 vtss_sync_api.h, 696
 vtss_sync_clock_out_t, 287
 divider, 287
 enable, 287
 vtss_sync_divider_t
 vtss_sync_api.h, 695
 vtss_tag_remark_mode_t
 vtss_qos_api.h, 678
 vtss_tag_type_t
 vtss_packet_api.h, 569
 vtss_target_type_t
 vtss_init_api.h, 441
 vtss_tci_t, 288

cfi, 288
tagprio, 289
vid, 288
vtss_temp_sensor_get
 vtss_misc_api.h, 535
vtss_temp_sensor_init
 vtss_misc_api.h, 534
vtss_timeofday_t, 289
 sec, 289, 290
vtss_timestamp_age
 vtss_ts_api.h, 713
vtss_timestamp_t, 290
 nanoseconds, 291
 sec_msb, 290
 seconds, 291
vtss_tod_get_ns_cnt
 vtss_misc_api.h, 534
vtss_tod_set_ns_cnt_cb
 vtss_misc_api.h, 534
vtss_trace_conf_get
 vtss_misc_api.h, 517
vtss_trace_conf_set
 vtss_misc_api.h, 517
vtss_trace_conf_t, 291
 level, 292
vtss_trace_group_t
 vtss_misc_api.h, 511
vtss_trace_layer_t
 vtss_misc_api.h, 511
vtss_trace_level_t
 vtss_misc_api.h, 512
vtss_ts_adjtimer_get
 vtss_ts_api.h, 704
vtss_ts_adjtimer_one_sec
 vtss_ts_api.h, 702
vtss_ts_adjtimer_set
 vtss_ts_api.h, 704
vtss_ts_api.h
 vtss_rx_master_timestamp_get, 712
 vtss_rx_timestamp_get, 711
 vtss_rx_timestamp_id_release, 712
 vtss_timestamp_age, 713
 vtss_ts_adjtimer_get, 704
 vtss_ts_adjtimer_one_sec, 702
 vtss_ts_adjtimer_set, 704
 vtss_ts_delay_asymmetry_get, 709
 vtss_ts_delay_asymmetry_set, 709
 vtss_ts_egress_latency_get, 708
 vtss_ts_egress_latency_set, 708
 vtss_ts_external_clock_mode_get, 705
 vtss_ts_external_clock_mode_set, 705
 vtss_ts_external_clock_saved_get, 706
 vtss_ts_freq_offset_get, 705
 vtss_ts_ingress_latency_get, 707
 vtss_ts_ingress_latency_set, 706
 vtss_ts_internal_mode_get, 711
 vtss_ts_internal_mode_set, 710
 vtss_ts_internal_mode_t, 294
 int_fmt, 294
 vtss_ts_ongoing_adjustment
 vtss_ts_api.h, 703
 vtss_ts_operation_mode_get
 vtss_ts_api.h, 710
 vtss_ts_operation_mode_set
 vtss_ts_api.h, 709
 vtss_ts_operation_mode_t, 295
 mode, 295
 vtss_ts_p2p_delay_get
 vtss_ts_api.h, 707
 vtss_ts_p2p_delay_set
 vtss_ts_api.h, 707
 vtss_ts_status_change
 vtss_ts_api.h, 713

vtss_ts_timeofday_get
 vtss_ts_api.h, 703
 vtss_ts_timeofday_next_pps_get
 vtss_ts_api.h, 703
 vtss_ts_timeofday_offset_set
 vtss_ts_api.h, 702
 vtss_ts_timeofday_set
 vtss_ts_api.h, 701
 vtss_ts_timeofday_set_delta
 vtss_ts_api.h, 701
 vtss_ts_timestamp_alloc_t, 295
 cb, 296
 context, 296
 port_mask, 296
 vtss_ts_timestamp_t, 296
 context, 297
 id, 297
 ts, 297
 ts_valid, 297
 vtss_tx_timestamp_idx_alloc
 vtss_ts_api.h, 713
 vtss_tx_timestamp_update
 vtss_ts_api.h, 711
 vtss_uc_flood_members_get
 vtss_l2_api.h, 493
 vtss_uc_flood_members_set
 vtss_l2_api.h, 493
 vtss_vcap_bit_t
 types.h, 400
 vtss_vcap_ip_t, 298
 mask, 298
 value, 298
 vtss_vcap_key_type_t
 types.h, 401
 vtss_vcap_u128_t, 299
 mask, 299
 value, 299
 vtss_vcap_u16_t, 300
 mask, 300
 value, 300
 vtss_vcap_u24_t, 301
 mask, 301
 value, 301
 vtss_vcap_u32_t, 302
 mask, 302
 value, 302
 vtss_vcap_u40_t, 303
 mask, 303
 value, 303
 vtss_vcap_u48_t, 304
 mask, 304
 value, 304
 vtss_vcap_u8_t, 305
 mask, 305
 value, 305
 vtss_vcap_udp_tcp_t, 306
 high, 307
 in_range, 306
 low, 306
 mask, 308
 value, 307
 vtss_vcap_vid_t, 307
 high, 309
 low, 309
 mask, 309
 r, 310
 type, 309
 v, 309
 value, 309
 vr, 310
 vtss_vcap_vr_t, 308
 high, 309
 low, 309
 mask, 309
 r, 310
 type, 309
 v, 309
 value, 309
 vr, 310
 vtss_vcap_vr_type_t
 types.h, 401
 vtss_vce_action_t, 310
 policy_no, 311
 vid, 310
 vtss_vce_add
 vtss_l2_api.h, 478
 vtss_vce_del
 vtss_l2_api.h, 478
 vtss_vce_frame_etype_t, 311
 data, 312
 etype, 311
 vtss_vce_frame_ipv4_t, 312
 dport, 313
 dscp, 313
 fragment, 312
 options, 313
 proto, 313
 sip, 313
 vtss_vce_frame_ipv6_t, 314
 dport, 315
 dscp, 314
 proto, 314
 sip, 315
 vtss_vce_frame_llc_t, 315
 data, 316
 vtss_vce_frame_snap_t, 316
 data, 316
 vtss_vce_init
 vtss_l2_api.h, 477
 vtss_vce_key_t, 317
 etype, 318
 frame, 319
 ipv4, 318
 ipv6, 319
 llc, 318
 mac, 317
 port_list, 317
 snap, 318
 tag, 317
 type, 318
 vtss_vce_mac_t, 319
 dmac_bc, 320
 dmac_mc, 320
 smac, 320

vtss_vce_t, 320
action, 321
id, 321
key, 321
vtss_vce_tag_t, 321
dei, 322
pcp, 322
s_tag, 323
tagged, 322
vid, 322
vtss_vce_type_t
 vtss_l2_api.h, 461
vtss_vcl_port_conf_get
 vtss_l2_api.h, 476
vtss_vcl_port_conf_set
 vtss_l2_api.h, 477
vtss_vcl_port_conf_t, 323
 dmac_dip, 323
vtss_vdd_t
 types.h, 400
vtss_vid_mac_t, 324
 mac, 324
 vid, 324
vtss_vlan_conf_get
 vtss_l2_api.h, 472
vtss_vlan_conf_set
 vtss_l2_api.h, 472
vtss_vlan_conf_t, 325
 s_etype, 325
vtss_vlan_frame_t
 types.h, 399
vtss_vlan_port_conf_get
 vtss_l2_api.h, 473
vtss_vlan_port_conf_set
 vtss_l2_api.h, 473
vtss_vlan_port_conf_t, 326
 frame_type, 327
 ingress_filter, 327
 port_type, 326
 pvid, 326
 untagged_vid, 326
vtss_vlan_port_members_get
 vtss_l2_api.h, 474
vtss_vlan_port_members_set
 vtss_l2_api.h, 474
vtss_vlan_port_type_t
 vtss_l2_api.h, 461
vtss_vlan_tag_t, 327
 dei, 328
 pcp, 328
 tpid, 328
 vid, 328
vtss_vlan_trans_group_add
 vtss_l2_api.h, 478
vtss_vlan_trans_group_del
 vtss_l2_api.h, 479
vtss_vlan_trans_group_get
 vtss_l2_api.h, 479
vtss_vlan_trans_group_to_port_get
 vtss_l2_api.h, 480
vtss_vlan_trans_group_to_port_set
 vtss_l2_api.h, 480
vtss_vlan_trans_grp2vlan_conf_t, 329
 group_id, 329
 trans_vid, 329
 vid, 329
vtss_vlan_trans_port2grp_conf_t, 330
 group_id, 330
 ports, 330
vtss_vlan_tx_tag_get
 vtss_l2_api.h, 475
vtss_vlan_tx_tag_set
 vtss_l2_api.h, 476
vtss_vlan_tx_tag_t
 vtss_l2_api.h, 461
vtss_vlan_vid_conf_get
 vtss_l2_api.h, 475
vtss_vlan_vid_conf_set
 vtss_l2_api.h, 475
vtss_vlan_vid_conf_t, 331
 learning, 331
 mirror, 331
vtss_vt_id_t
 vtss_l2_api.h, 460
vtss_wol_mac_addr_t, 332
 addr, 332
vtss_wol_passwd_len_type_t
 vtss_phy_api.h, 615
warm_start_enable
 vtss_init_conf_t, 113
width
 vtss_pi_conf_t, 213
wol_mac
 vtss_phy_wol_conf_t, 211
wol_pass
 vtss_phy_wol_conf_t, 212
wol_passwd_len
 vtss_phy_wol_conf_t, 212
xaui_rx_lane_flip
 vtss_port_conf_t, 222
xaui_tx_lane_flip
 vtss_port_conf_t, 222
xtr_cb
 vtss_fdma_ch_cfg_t, 103
xtr_grp
 vtss_fdma_ch_cfg_t, 102
xtr_qu
 vtss_packet_rx_meta_t, 160
xtr_qu_mask
 vtss_packet_rx_info_t, 154